MICROCOMPUTER I/O FOR A REAL-TIME AUTOMATIC EQUALIZER

By

James Lee Stortz
B.S., University of Louisville 1984

A Thesis
Submitted to the Faculty of the
University of Louisville
Speed Scientific School
as Partial Fulfillment of the Requirements
for the Professional Degree

MASTER OF ENGINEERING

Department of Electrical Engineering

December 1985

MICROCOMPUTER I/O FOR A
REAL-TIME AUTOMATIC EQUALIZER

Submitted by: James L. Stortz
James Lee Stortz

A Thesis Approved on

October 1, 1985
Date

by the Following Reading and Examination Committee:

J. Rodney Cox
Thesis Director, J. Rodney Cox

Jacek Zurada
J. M. Zurada

Thomas L. Holloman
T. L. Holloman

ACKNOWLEDGEMENTS

The author wishes to extend his appreciation to J. Rodney Cox, Assistant Professor of Electrical Engineering for his counseling and advice contributing to the development of this thesis.

The author also wishes to express gratitude toward his parents, Patricia and Philip Stortz, for their encouragement as well as financial support.

Finally, the author is grateful for the assistance and support of his wife, Ellen Stortz.

ABSTRACT

This paper focuses on the development of I/O capabilities for an Automatic Real-Time Equalizer. The excellent graphics capabilities of the Commodore 64 are utilized to provide a high-resolution display showing the operation of the equalizer. Input operations are accomplished with the use of a simple menu. A unique raster scan interrupt technique is presented that allows the menu to take the place of a portion of the display. A keyboard interrupt routine allows for menu selection while the display is active.

TABLE OF CONTENTS

NOMENCLATURE

ADSP	Adaptive Digital Signal Processor
A/D	Analog-to-Digital Converter
BCD	Binary Coded Decimal
CRT	Cathode Ray Tube
dB	Decibels
D/A	Digital-to-Analog Converter
DCA	Digitally Controlled Amplifiers
Hz	Hertz
$	Hexadecimal Number
KHz	Kiloherz
LED	Light Emitting Diode
PLL	Phase Locked Loop
RTA	Real-Time Analyzer
SPL	Sound Pressure Level
VIC	Video Interface Chip
VCO	Voltage Controlled Oscillator

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Today, sound-system equalization is a widely accepted and applied practice. Equalization is used to produce improvements in sound system naturalness, intelligibility, and realism, as well as, increases in gain before feedback in sound reinforcement systems. A sound reinforcement system is any system used to sense, amplify, and present audio programming to an audience (Figure 1a).

To properly set up an equaliztion system, a person knowledgeable with acoustics is required. The process for determining the equalizer´s settings is generally time-consuming and tedious. Furthermore, once the system is properly equalized, a trained operator is necessary to make adjustments in the equalizer settings to compensate for the rapidly changing conditions that arise during the sound system´s usage. With some type of automatic control, these disadvantages can be overcome.

With the system described herein, equalization is performed quickly and efficiently. Once equalization is complete, the automatic equalizer continues to monitor the situation and take corrective action against the development of unwanted feedback.

This thesis contributes to the development of the automatic equalizer by providing input and output operations. The output takes the form of a high-resolution graphics

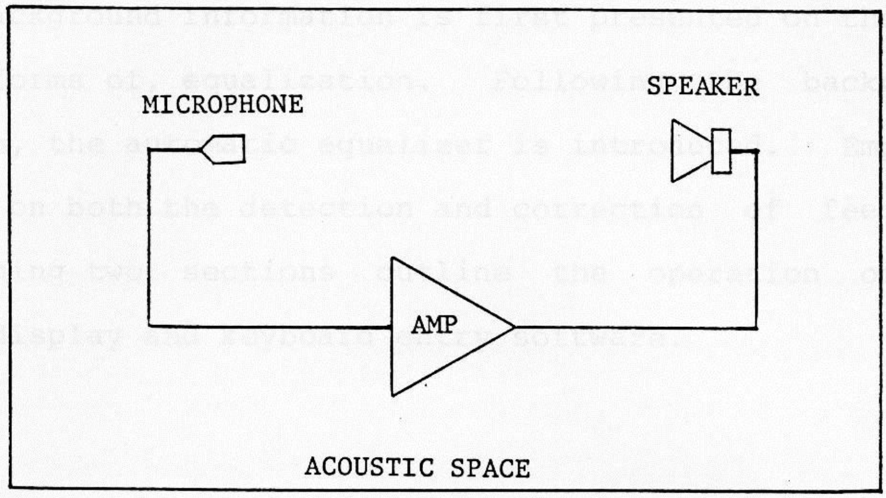

FIGURE 1a - Sound Reinforcement System

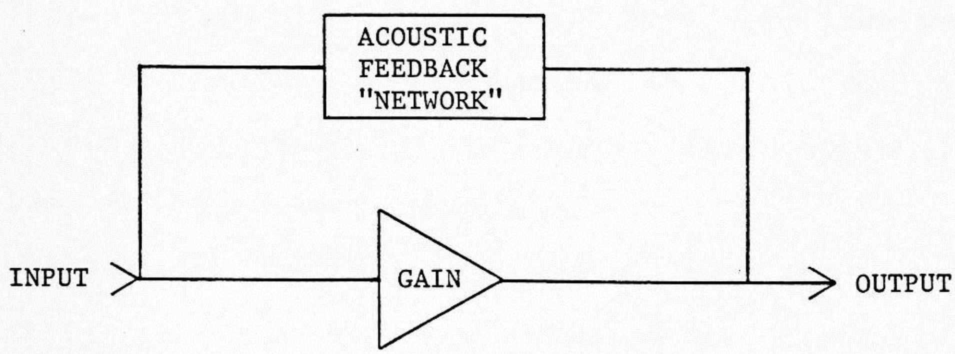

FIGURE 1b - Model of a Sound Reinforcement System

display. This display shows the status of each operation as it is performed. The input function allows the operation of the automatic equalizer to be modified manually.

Background information is first presented on the need for, and forms of, equalization. Following the background discussion, the automatic equalizer is introduced. Emphasis is placed on both the detection and correction of feedback. The remaining two sections outline the operation of the graphics display and keyboard entry software.

II. BACKGROUND

In a sound reinforcement system, there is always some acoustic coupling between the loudspeaker and the microphone so that the entire system forms a closed loop. This coupling is shown in Figure 1b as an acoustic feedback "network". The acoustic "network" is similar to an electrical network in that a certain response in produced by a particular input. The response produced is determined by the physical characteristics of the room. Speaker location, the number and location of people in the room, furnishing placements and construction materials all enter into this complex response.

In order for the system to remain stable, that is, not go into oscillation, it is necessary that at any frequency for which the overall phase shift is an integral multiple of 360 degrees, the loop gain must be less than one.[1] Whenever this stability criterion is violated, the system will oscillate, or "ring" at those frequencies for which the loop gain is greater than one.

The amplitude response for a system that meets the stability criterion, will have an absence of frequencies which have amplitudes significantly greater than the others, as illustrated in Figure 2a. Once instability occurs, the response of the oscillatory frequency can "swell", ultimately becoming much greater in level that the program signal.[2] This condition is depicted by the narrow peak in Figure 2b.

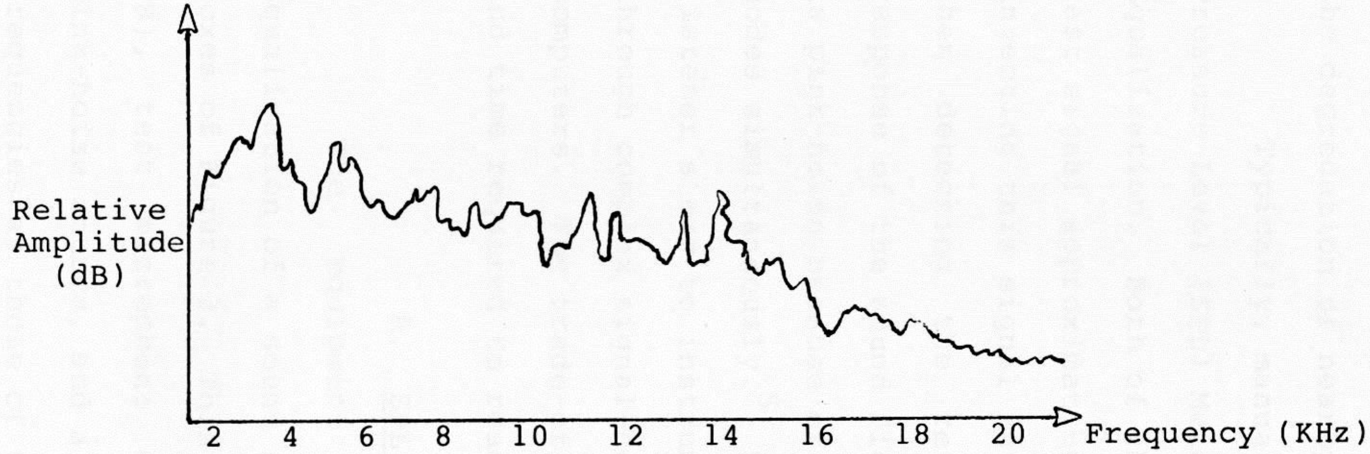

FIGURE 2a - Spectrum Display Of Music Without Any Oscillations

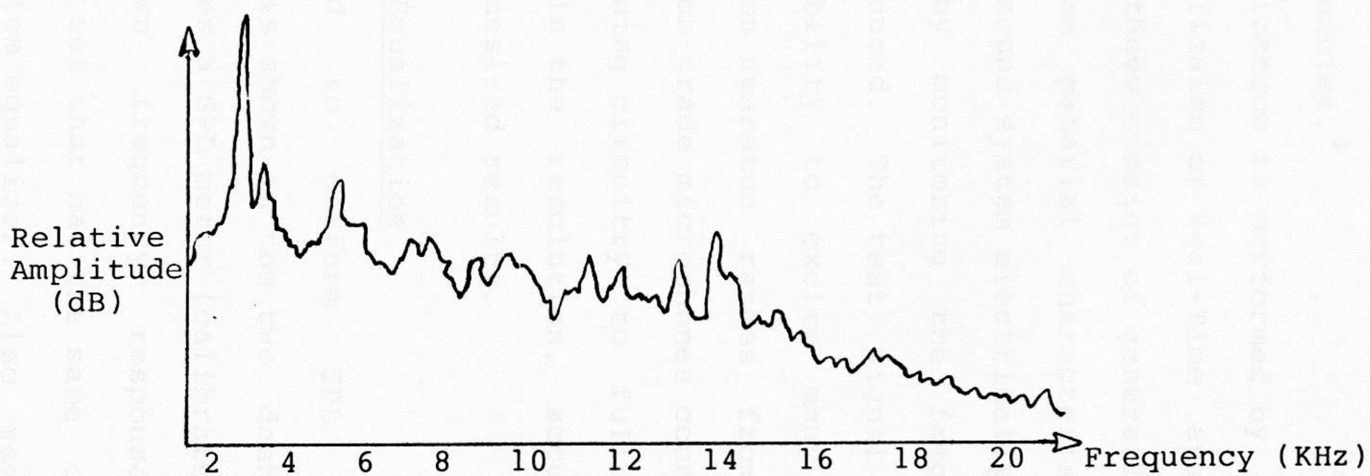

FIGURE 2b - Spectrum Display Of Music With An Oscillation Present

In most applications, acoustic feedback occurrence is controlled by smoothing the room response with some sort of equalizer.[3] The preferred type is the one-third octave equalizer because it is recognized as having enough frequency resolution for the attenuation of problem frequencies without the degredation of nearby frequencies.[4]

Typically, manual equalization is performed by Sound Pressure Level (SPL) Meter Equalizaion or Real-Time analyzer Equalization. Both of these methods consist of generating a test signal approximating program material characteristics, injecting this signal into the sound system electrically, and then detecting the results by monitoring the frequency response of the sound field produced. The test signal used is pink-noise because of its ability to excite many room modes simultaneously.[5] Detection aparatus ranges from the listener´s ear to instrumentation-grade microphones connected through complex signal-conditioning circuitry to full-scale computers. The trade-offs lie in the resolution, accuracy, and time required to reach the desired results.

A. SPL Meter Equalization

The equipment needed to perform SPL Meter Equalization of a sound system is shown in the two dashed-in boxes of Figure 3. This includes a SPL meter (calibrated in dB), test microphone of known frequency response, a pink-noise source, and a filter set that has the same center frequencies as those of the system equalizer. Also required of the filter set, is the ability to turn filters on and off

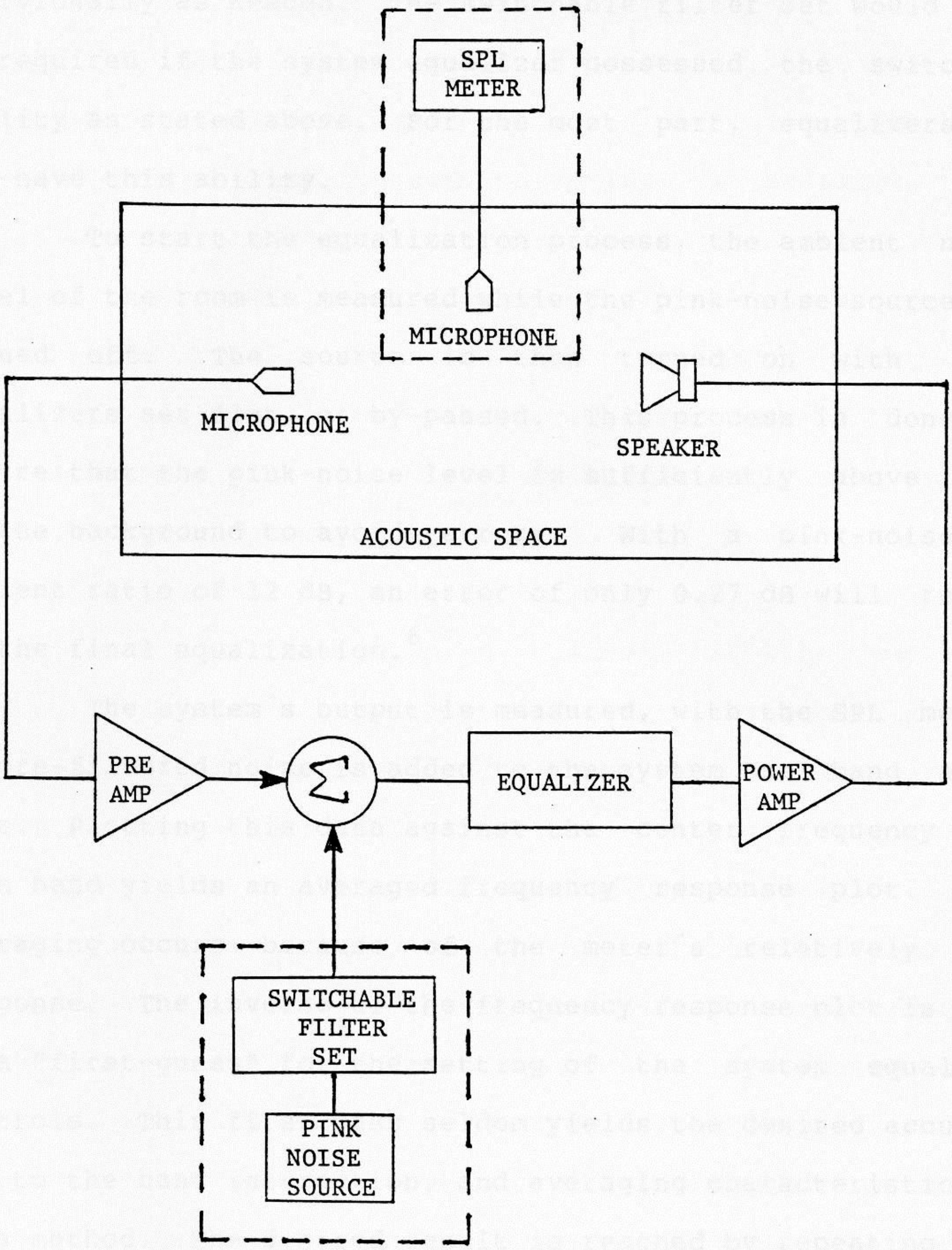

FIGURE 3 - SPL Meter Equalization of a Sound Reinforcement System

individually as needed. The switchable filter set would not be required if the system equalizer possessed the switching ability as stated above. For the most part, equalizers do not have this ability.

To start the equalization process, the ambient noise level of the room is measured while the pink-noise source is turned off. The source is then turned on with both equalizers set flat, or by-passed. This process is done to ensure that the pink-noise level is sufficiently above that of the background to avoid errors. With a pink-noise to ambient ratio of 12 dB, an error of only 0.27 dB will result in the final equalization.[6]

The system´s output is measured, with the SPL meter, as pre-filtered noise is added to the system one band at a time. Plotting this data against the center frequency for each band yields an averaged frequency response plot. The averaging occurs because of the meter´s relatively slow response. The inverse of the frequency response plot is used as a "first-guess" for the setting of the system equalizer controls. This first pass seldom yields the desired accuracy due to the band interaction, and averaging characteristics of this method. The desired result is reached by repeating the process in an iterative manner.

It is this iterative process that makes this method time-consuming and tedious, but this is just the sort of process that machine language programs perform well. As will be shown, there are several devices that use this process.

B. Real-Time Analyzer Equalization

The real-time analyzer (RTA) essentially consists of a group of level detector/averaging circuits that are preceded by band-pass filters operating in parallel. The output of the level detector/averaging stages drive a readout that provides a continuous display of the frequency response of the sound system. The display, most often made of a matrix of LED´s, shows the response of each band simultaneously, making the determination of problem frequencies during equalization easy.

The set-up for using the RTA for equalizing a sound system is shown in Figure 4. Note that the only two components needed are the RTA, and the pink-noise source. Equalization progresses as follows. The master gain of the system is increased until the system is within a few dB of oscillation. The system is "ridden" close to oscillation to ensure that the equalization process takes care of frequencies that tend to "swell" in amplitude near feedback. Once at this point, the system equalizer is adjusted until the RTA display is flat, or of the desired shape. It should be noted that a flat frequency response appears a bit "bright" to the average listener. A more pleasing sound is obtained if the response is made to roll-off at three to six dB per octave with increasing frequency.[7]

The main advantage in using the RTA is that each adjustment of the equalizer is literally seen on the screen of the analyzer. Thus, the tedium of using an iterative process as with the SPL meter method is eliminated.

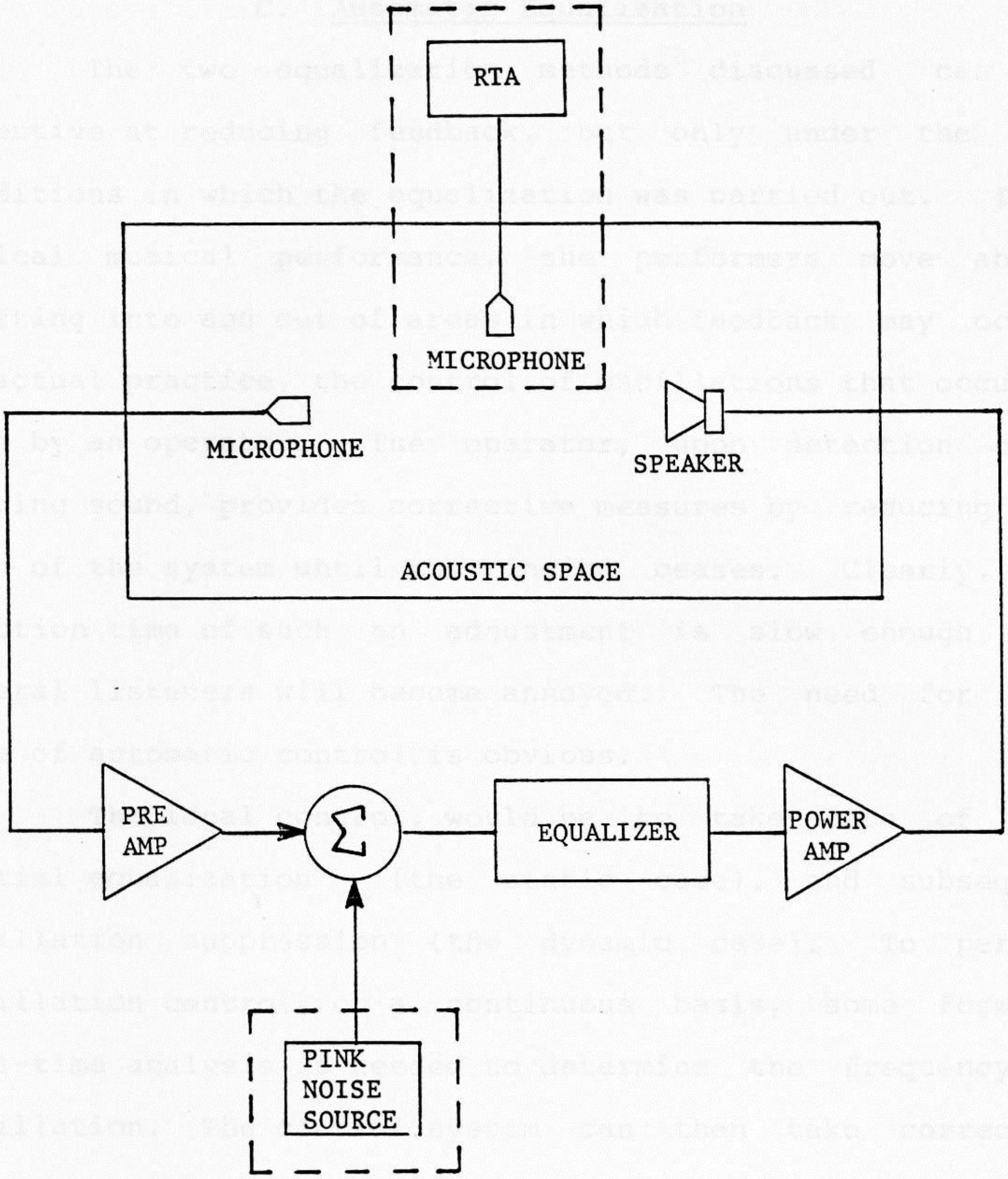

FIGURE 4 - RTA Equalization of a Sound Reinforcement System

C. Automatic Equalization

The two equalization methods discussed can be effective at reducing feedback, but only under the same conditions in which the equalization was carried out. In a typical musical performance, the performers move about, drifting into and out of areas in which feedback may occur. In actual practice, the control of oscillations that occur is done by an operator. The operator, upon detection of a ringing sound, provides corrective measures by reducing the gain of the system until the ringing ceases. Clearly, the reaction time of such an adjustment is slow enough that several listeners will become annoyed. The need for some type of automatic control is obvious.

The ideal control, would be to take care of both initial equalization (the static case), and subsequent oscillation suppression (the dynamic case). To perform oscillation control, on a continuous basis, some form of real-time analysis is needed to determine the frequency of oscillation. The control system can then take corrective measures, such as lowering the gain of the appropriate band of the equalizer.

Although not widely used, there are systems on the market that perform automatic equalization and/or oscillation suppression in real-time. The following is a look at some of these systems.

Eugene Patronis, Jr.[8] of the Georgia Institute of Technology has developed an oscillation suppression system that uses phase locked loops (PLL) for detection. An array

of PLL´s are arranged to cover the audio spectrum. When the array is fed an audio signal in which an oscillation is present, at least one of the PLL´s will lock to the frequency of oscillation. When a PLL is locked to a frequency, a constant error voltage output is generated. The error voltage is used to gate a COUNT UP clock signal to a BCD up/down counter. The output of the counter controls the master gain of the sound system through a circuit which uses a multiplying digital-to-analog converter configured as a digitally controlled amplifier. As the counter is incremented (oscillation detected) the gain is lowered until the PLL loses its lock, thus disabling the up count. The gain is gradually restored to its original value by a COUNT DOWN clock. This clock signal has a period much longer than the COUNT UP clock.

Patronis[9] has also proposed an oscillation detection scheme that is based on a correlation approach. By sampling a microphone input and comparing this input over time for correlation, oscillations can be detected. This is based on the observation that audio program material has rapidly varying frequency components. Any frequency components that persist over the sampling interval would be the result of oscillations.

This approach would require the use of a time-to-frequency domain conversion, such as the Fast Fourier Transform, and a large amount of computer memory. The time required for detection could possibly happen at a rate that would not allow for proper removal of the oscillations

without annoyance to several listeners.

Acoustic Research,[10] the well-known speaker system manufacturer, has developed an equalization system called the Adaptive Digital Signal Processor (ADSP). The ADSP works in the Time Domain to recognize and automatically correct frequency response errors in a sound system. The heart of the system is the 16-bit Texas Instruments TMS-9995 microcomputer chip. Since the ADSP has no preset filter bands, it can synthesize as many digital filters as needed for equalization. Typically, the ADSP will provide more than 50 extremely narrow band filters in a one KHz bandwidth. Because of Audio Research's belief that a flat frequency response above one KHz is mainly a function of the speaker design, the ADSP works only in the frequency region below one kHz.

Robert W. Adams[11] of the dbx Company has developed an automatic equalizer for home consumer use that is based on the real-time analyzer approach. The system uses a microprocessor to digitally adjust the band gains of an octave graphic equalizer based on data from an analyzer. There are manual controls for the band gains so that custom equalization curves can be obtained. The dbx system also has the ability to store and recall multiple equalizaion curves for use at any time.

A hybrid equalizing system is used by C. R. Guarino,[12] an independent audio consultant, in his automatic equalizer. In this design, the output of a calibrated microphone in the audience area is digitized and processed by

a microcomputer. The microcomputer determines the gain for each band of the equalizer needed for equalization and generates an analog control signal for the equalizer. The equalizer´s band gains are adjusted by voltage control means.

III. AUTOMATIC EQUALIZER

The real-time automatic equalizer discussed here has the ability to perform both automatic equalization prior to sound system usage, and oscillation suppression during usage. A block diagram showing the main functional divisions of the hardware is shown in Figure 5. In this arrangement, the audio signal is merely manipulated by computer controlled analog devices such as filters and op-amps. The overwhelming advantage of this approach is the absence of a need for a powerful computer, as would be required by a system based on a digitizing scheme. As will be discussed, a simple eight-bit microprocessor-based control system is more than sufficient.

A. Oscillation Detection

The method used to detect oscillations is an expansion on Patronis´ PLL philosophy. The automatic equalizer´s detection system is able to determine the frequency of oscillation, not just the mere presence of an oscillation as with Patronis´ system. Before discussing the detection system, a brief explanation of PLL operation is needed.

The basic diagram of a phase locked loop is shown in Figure 6. The closed-loop operation of the PLL is to maintain the voltage controlled oscillator (VCO) frequency locked to that of the input signal frequency. With no input

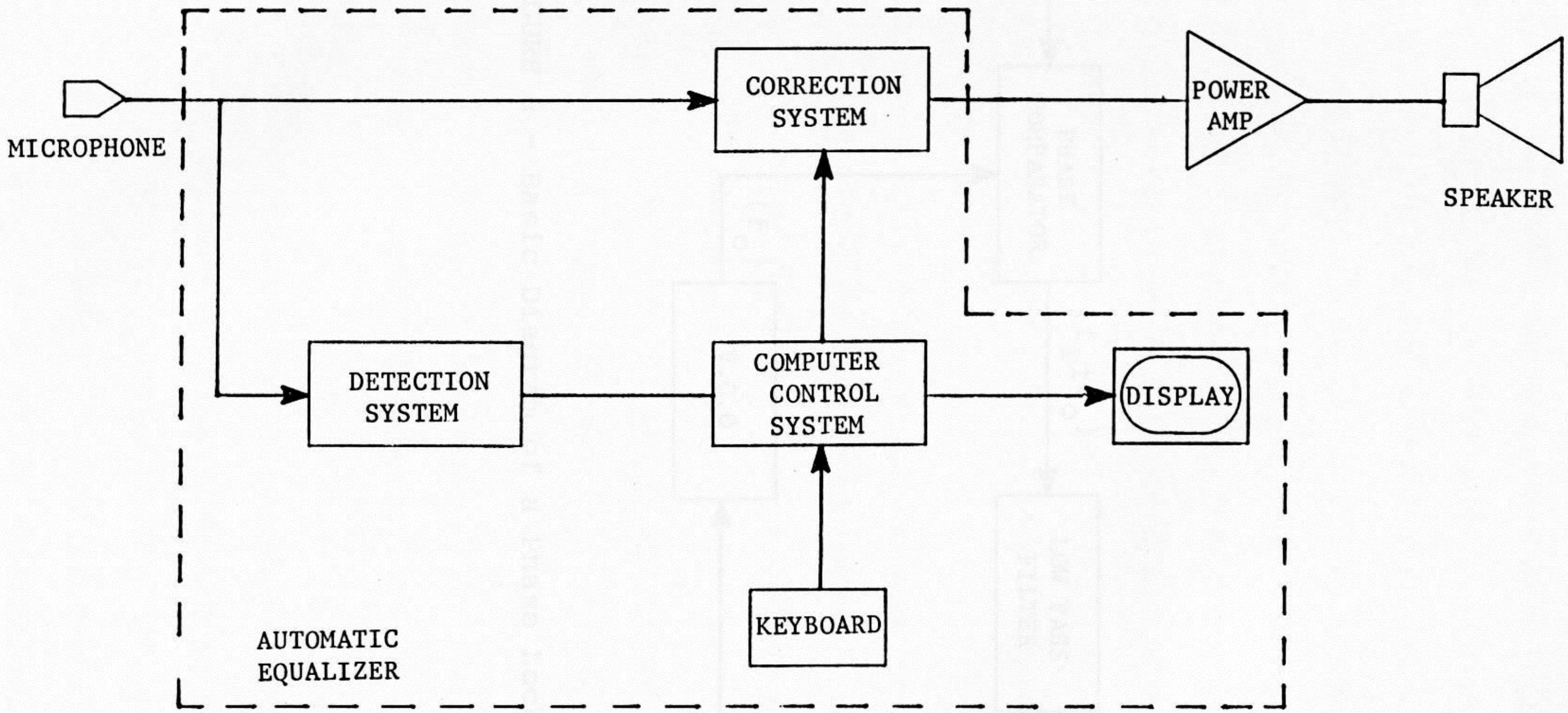

FIGURE 5 - Automatic Equalizer System

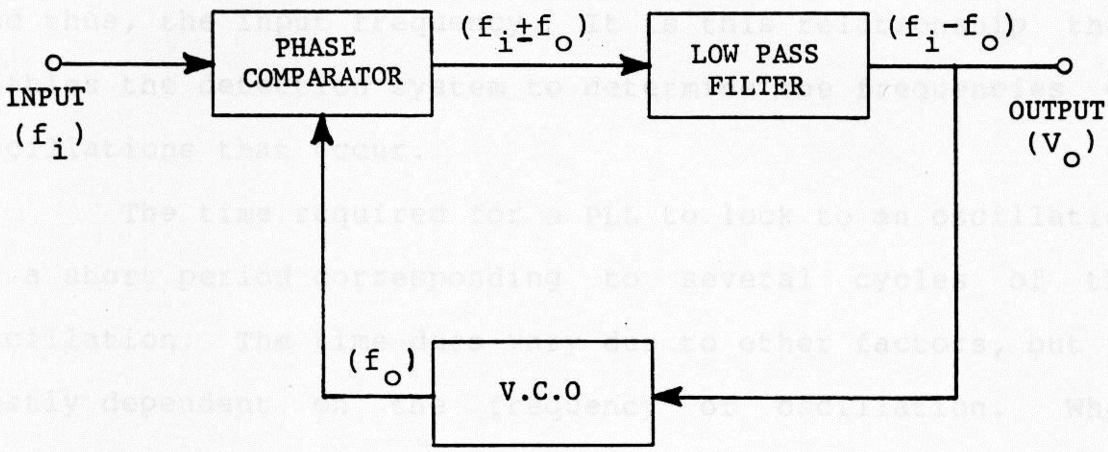

FIGURE 6 - Basic Diagram of a Phase Locked Loop

signal present, the VCO is at its rest, or center, frequency, f_o. With an input signal of frequency f_i present, a fixed DC voltage, v_o, is generated by the phase detector and low-pass filter. This voltage, which corresponds to the difference between f_i and f_o, is the value needed to hold the VCO in lock with the input. For most PLL's, the relationship between v_o and f_i is fairly linear over the frequency range of the PLL. By knowing the center frequency and the output voltage, it is possible to determine the frequency of the VCO and thus, the input frequency. It is this relationship that enables the detection system to determine the frequencies of oscillations that occur.

The time required for a PLL to lock to an oscillation is a short period corresponding to several cycles of the oscillation. The time does vary due to other factors, but is mostly dependent on the frequency of oscillation. When applied a signal consisting of music or speech, the PLL will try to lock onto the many decaying frequencies, but will never lock to any particular frequency for any significant period of time.[13] When oscillations are present, the PLL will lock to the frequency of oscillation, treating the music information as noise.

The detection hardware arrangement is shown in Figure 7. Enough PLL's are used to cover the entire audio spectrum (20 Hz - 20 kHz). The frequency ranges of the PLL's are arranged to overlap one anothers rest frequencies. This is done to ensure that oscillation occurring at the rest frequency of one PLL will be detected by another.

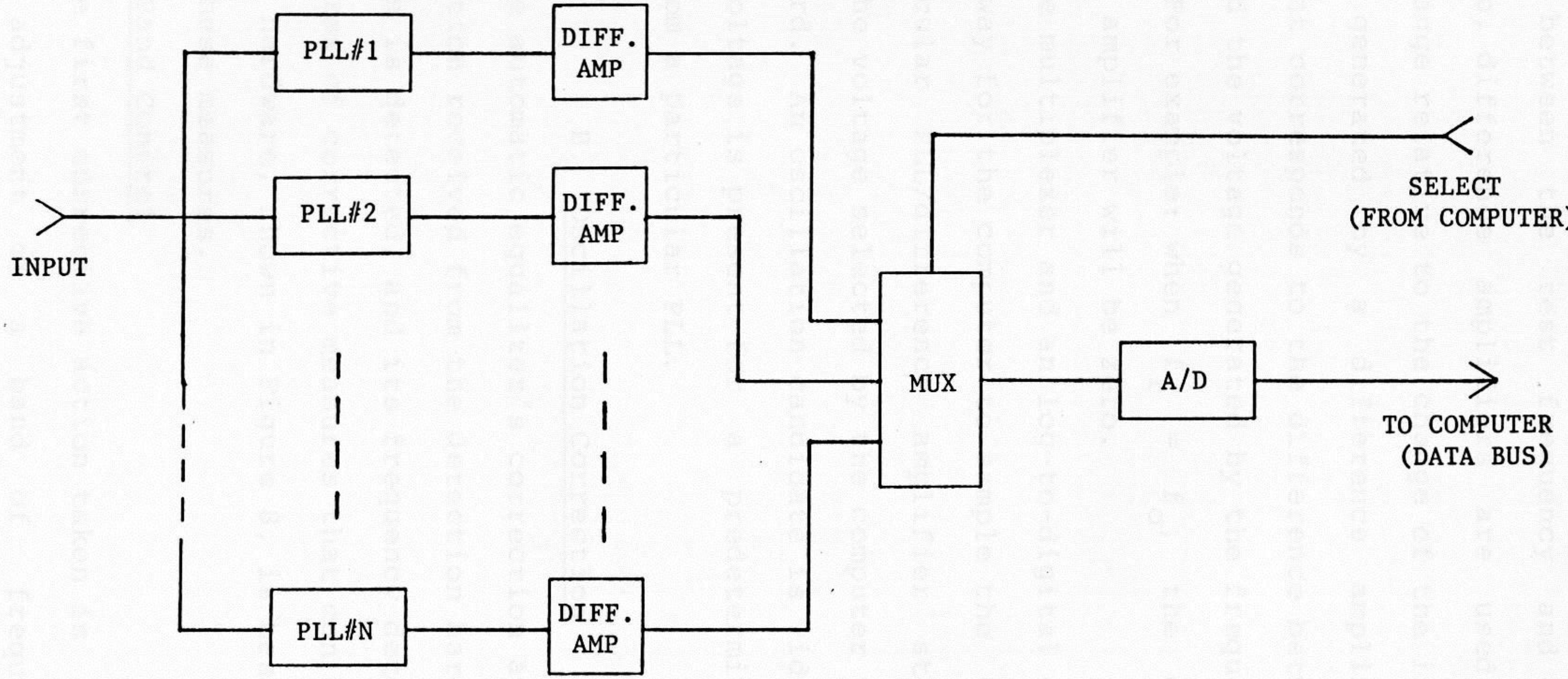

FIGURE 7 - PLL Hardware Arrangement

Since the output voltage of each PLL, represents the difference between the rest frequency and the frequency locked onto, difference amplifiers are used to make the voltage change relative to the change of the input frequency. The output generated by a difference amplifier is a DC voltage that corresponds to the difference between the rest voltage and the voltage generated by the frequency difference $f_i - f_o$. For example: when $f_i = f_o$, the output of the difference amplifier will be zero.

The multiplexer and analog-to-digital converter (A/D) provide a way for the computer to sample the output voltage of a particular PLL/difference amplifier stage. The A/D converts the voltage selected by the computer to an eight bit digital word. An oscillation candidate is identified if a constant voltage is present for a predetermined number of samples from a particular PLL.

B. Oscillation Correction

The automatic equalizer's correction approach relies on information received from the detection hardware. Once an oscillation is detected, and its frequency determined, there are two forms of corrective measures that can be taken. The correction hardware, shown in Figure 8, is best presented in terms of these measures.

1. Broad-Band Control

The first corrective action taken is concerned with the gain adjustment of a band of frequencies. This broad-band approach is done with an equalizer. The bandpass

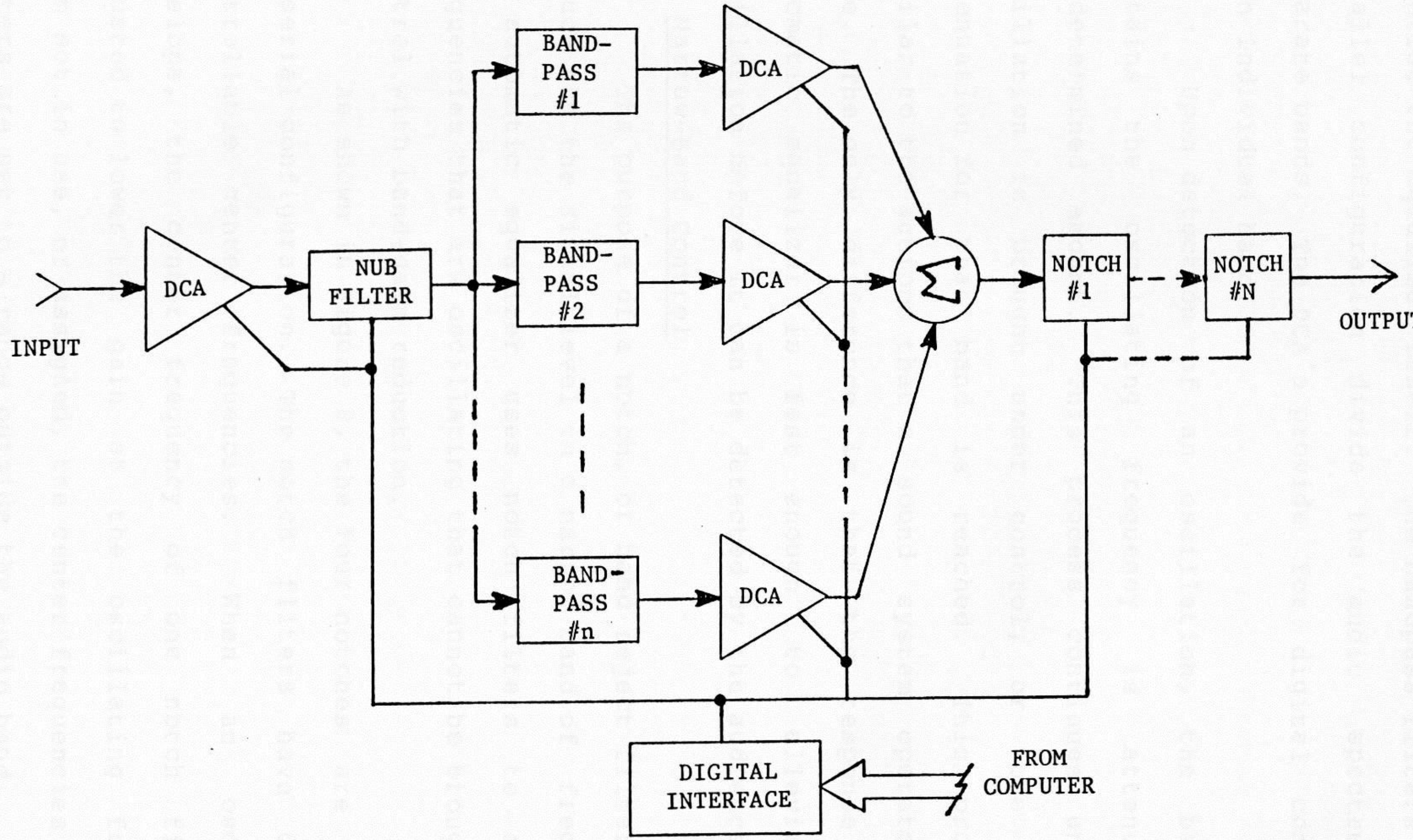

FIGURE 8 - Correction Hardware System

filters and the digitally controlled amplifiers (DCA) comprise the equalizer itself. The bandpass filters in this parallel configuration divide the audio spectrum into separate bands. The DCA´s provide for digital control of each individual band.

Upon detection of an oscillation, the band that contains the oscillating frequency is attenuated a predetermined amount. This process continues until the oscillation is brought under control, or the maximum attenuation for that band is reached. This process is similar to the actions that a sound system operator would take. The only difference is that the response of the automatic equalizer is fast enough to alleviate the oscillation before it can be detected by the audience.

2. Narrow-Band Control

The purpose of a notch, or band reject filter, is to reduce the the signal level in a narrow band of frequencies. The automatic equalizer uses notch filters to attenuate frequencies that are oscillating that cannot be brought under control with band-gain reduction.

As shown in Figure 8, the four notches are arranged in serial configuration. The notch filters have digitally controllable center frequencies. When an oscillation develops, the center frequency of one notch filter is adjusted to lower the gain at the oscillating frequency. When not in use, or assigned, the center frequencies of the filters are set to a range outside the audio band.

C. Computer Control

The computer control system has the task of coordinating both the functions of automatic equalization and oscillation suppression. These functions are performed by analyzing data from the PLL detection system and then directing the appropriate control to the bandpass and notch filters.

1. Automatic Equalization

As previously discussed, the goal of equalization is to provide a flat frequency response across the audio spectrum. This is accomplished by increasing or decreasing the gain of each bandpass filter stage of the equalizer. This will compensate for the variations in room geometry and other physical factors.

The process used by the automatic equalizer to perform equalization is very similar to the time proven SPL Meter process. The flow chart of Figure 9 illustrates the the automatic equalization process. First, a pink-noise source is turned on to generate energy in all frequency bands. The master gain of the equalizer is increased until an oscillation is detected. Once an oscillation is detected, the corresponding frequency band of the equalizer is attenuated until the oscillation ceases.

After the oscillation is squelched, the master gain is again increased until another oscillation is detected. This process is reiterated until all bands of the equalizer have been adjusted, or the maximum level of the master gain is reached. Ideally, the last increase of the master gain

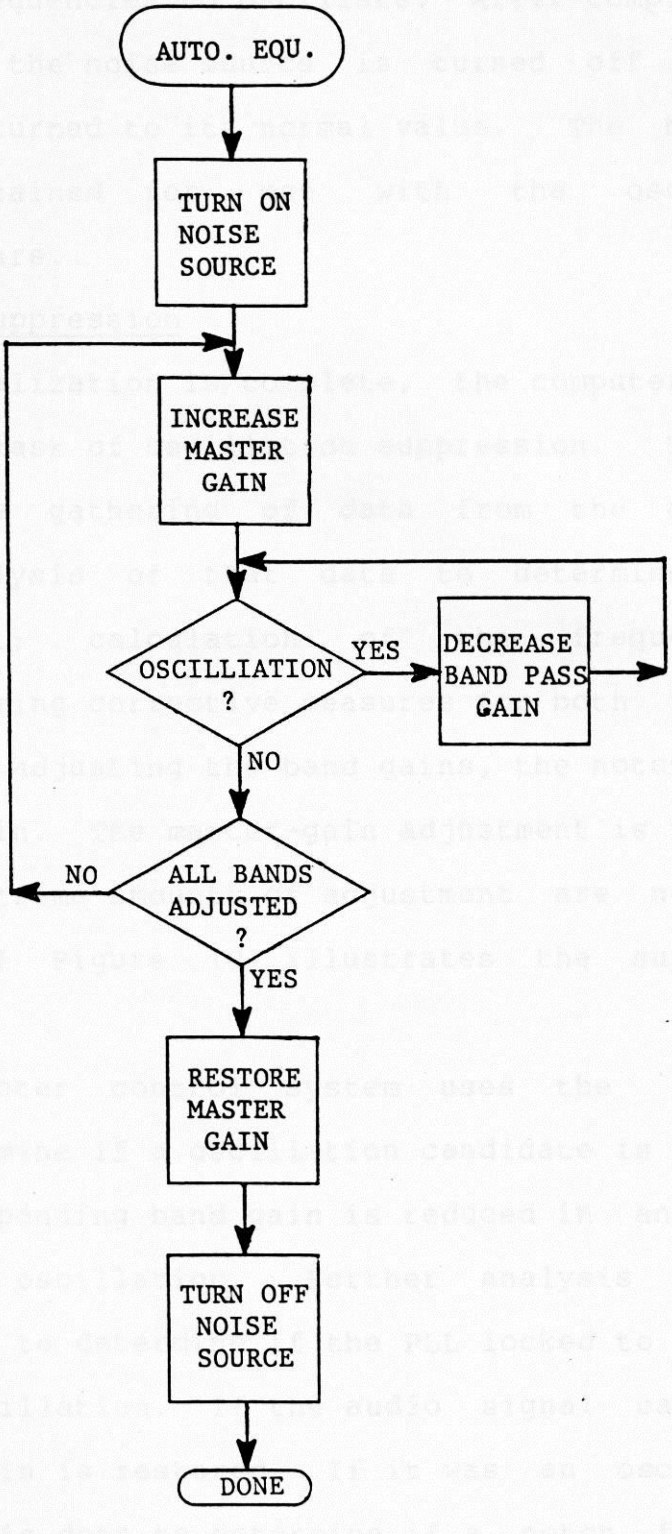

FIGURE 9 - Automatic Equalization Flow Chart

will cause all frequencies to oscillate. After completion of the equalization, the noise source is turned off and the master gain is returned to its normal value. The band-gain settings are retained for use with the oscillation suppression software.

2. Oscillation Suppression

After equalization is complete, the computer control switches to the task of oscillation suppression. The task involved are: the gathering of data from the detection hardware; the analysis of that data to determine if an oscillation exist; calculation of the frequency of oscillation; starting corrective measures for both the long and short term by adjusting the band gains, the notch filters and the master gain. The master-gain adjustment is used only in cases where extreme amounts of adjustment are necessary. The flow chart of Figure 10 illustrates the suppression approach.

The computer control system uses the detection hardware to determine if a oscillation candidate is present. If so, the corresponding band gain is reduced in an attempt to control the oscillation. Further analysis of the candidate is done to determine if the PLL locked to the audio signal, or an oscillation. If the audio signal caused the lock, the band gain is restored. If it was an oscillation, further analysis is done to determine if a notch filter is needed before the band gain can be restored.

Before a notch filter is assigned, a check is made to determine if any filters are available. If all the filters

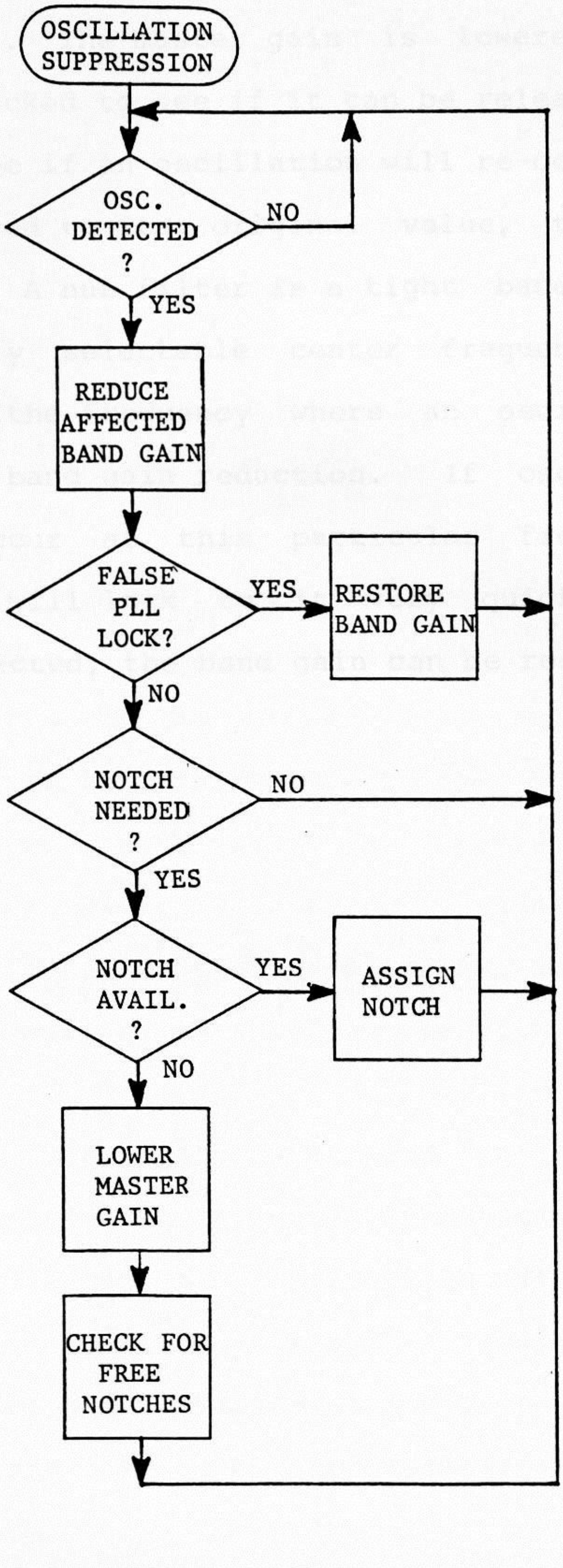

FIGURE 10 - Oscillation Suppression Flow Chart

are assigned, the condition arises where the master gain needs to be lowered. The master gain is lowered and each notch filter is checked to see if it can be released.

To determine if an oscillation will re-occur should a band gain be restored to its original value, the computer uses a nub filter. A nub filter is a tight bandpass filter that has a digitally selectable center frequency. A nub filter is moved to the frequency where an oscillation was suppressed through band gain reduction. If oscillation is still likely to occur at this particular frequency, the detection hardware will lock to it very quickly. If no oscillation is detected, the band gain can be restored to its original value.

IV. GRAPHICS DISPLAY

Although the automatic equalizer is capable of stand-alone operation, most sound system operators would prefer the flexibility of some form of manual control. To accommodate this, a high-resolution graphics display along with a keyboard entry scheme was developed. A Commodore 64 microcomputer was selected to provide both of these functions.

The display is used to show, in real-time, the status of each operation that the automatic equalizer is performing. To do this, the CRT screen is broken into separate fields (displays) to provide information about the following: the initial equalization settings; the present band-gain settings; the present energy spectrum; the availability and the frequency of assignment for all the notch and nub filters. This arrangement is shown in the drawing of Figure 11.

As shown, bar-graphs are used to represent the settings for the initial equalization, the band gains and the energy spectrum. The initial equalization display is the only display that remains constant. Changes in the equalization setting are reflected in the band-gains display. As the band gains are altered by the equalizer to accommodate changes in the acoustical environment, the corresponding bar-graph heights are updated on the display. The energy spectrum display operates in the same manner, except that

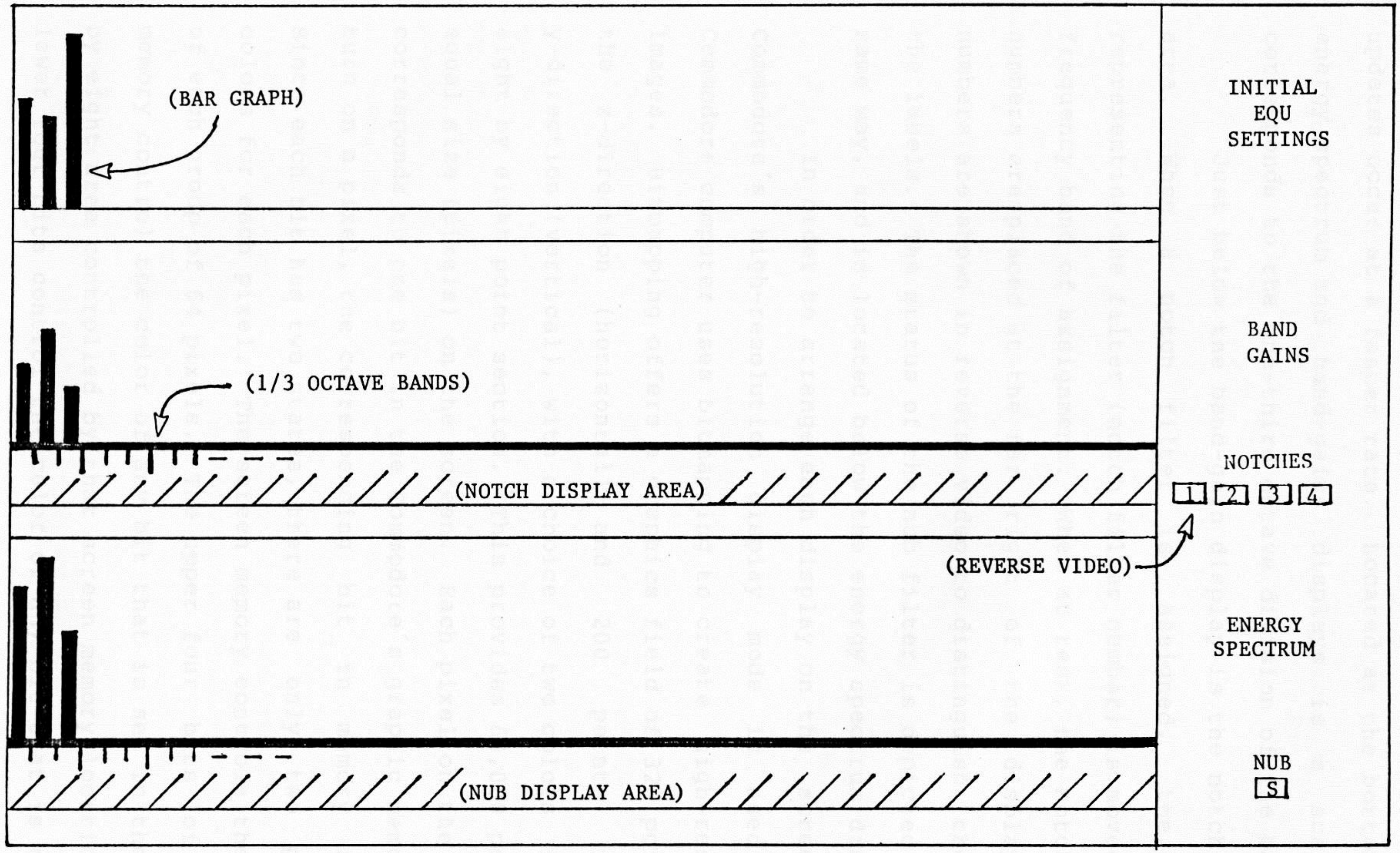

FIGURE 11 - Graphics Display

updates occur at a faster rate. Located at the bottom of the energy spectrum and band-gains displays is a scale that corresponds to the one-third octave division of the bands.

Just below the band-gain display is the notch display area. When a notch filter is assigned, the number representing the filter (notch filter number) is moved to the frequency band of assignment. When at rest, the notch filter numbers are placed at the far right of the display. The numbers are shown in reverse video to distinguish them from the labels. The status of the nub filter is depicted in the same way, and is located below the energy spectrum display.

In order to arrange each display on the screen, the Commodore´s high-resolution display mode is used. The Commodore computer uses bitmapping to create high-resolution images. Bitmapping offers a graphics field of 320 points in the x-direction (horizontal) and 200 points in the y-direction (vertical), with a choice of two colors in each eight by eight point section. This provides 64,000 points of equal size (pixels) on the screen. Each pixel on the screen corresponds to one bit in the Commodore´s graphic memory. To turn on a pixel, the corresponding bit in memory is set. Since each bit has two states, there are only two possible colors for each pixel. The screen memory controls the color of each group of 64 pixels. The upper four bits of screen memory control the color of any bit that is set in the eight by eight area controlled by that screen memory location. The lower four bits control the color of any bit that is reset.

To create the graphics display with the speed and resolution needed, machine language programs are used. For discussion purposes, the software is divided into two groups. The first is concerned with creating the initial high-resolution picture. The second group performs the operations needed to create the moving images. To facilitate ease of debugging and modification, the program is divided into small modules. Module interaction is kept to a minimum by using common memory tables. This programming structure is used so the display can be custom-tailored for specific applications without the need to develop an entirely new display program.

A. Initialization Software

The sequence used in creating the initial high-resolution image is shown in the flow chart of Figure 12. The programs that perform these operations are listed in Appendix B. The first task is to clear the bitmap memory. The routine MAPCLEAR does this by writing zeroes into each location of the bitmap RAM (random access memory). This RAM extends from $2000 to $3F3F.

Next, the color for each eight by eight group of pixels is set. This is performed in two separate stages. First, the SETCOLOR1 routine sets the entire screen to have a cyan color for the cleared pixels (background color) and a black color for the set pixels. This color combination works best for both black and white and color monitors. The second routine SETCOLOR2 fills in three horizontal bars with a dark blue color. These bars are used to partition the screen into

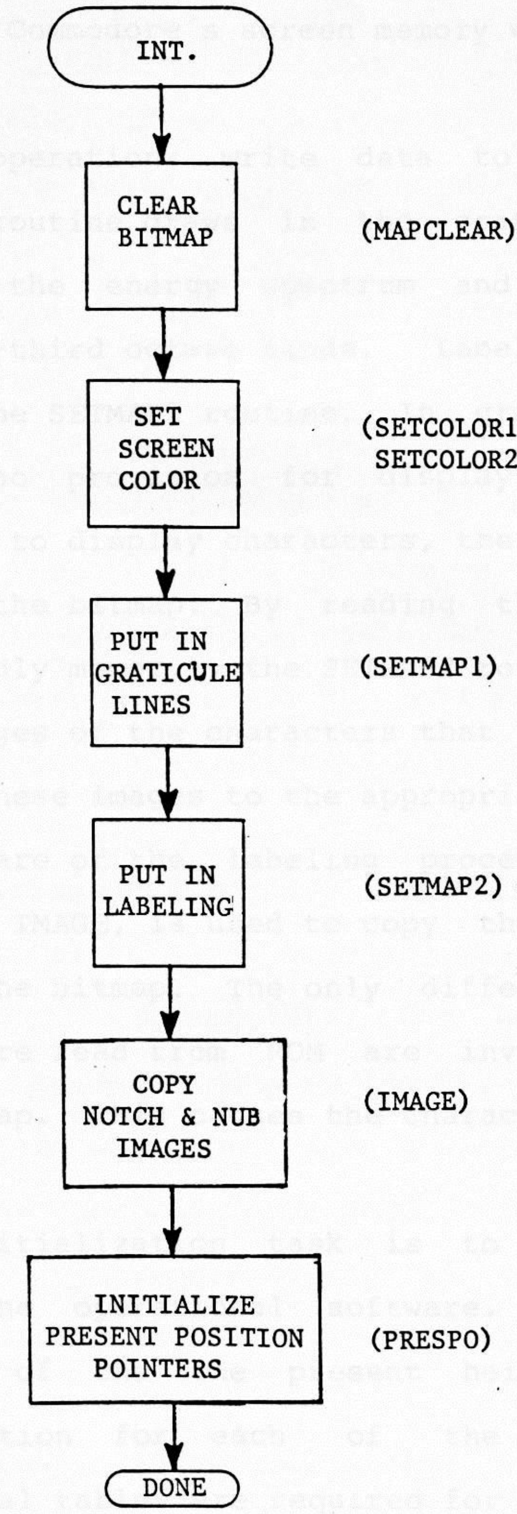

FIGURE 12 - Initialization Flow Chart

three large blocks. Both these programs set the color by writing data into the Commodore´s screen memory which extends from \$0400 to \$07E7.

The next two operations write data to the bitmap itself. The SETMAP1 routine draws in the graticule lines that that partition the energy spectrum and band gains displays into the one-third octave bands. Labeling of the screen is done with the SETMAP2 routine. In graphics mode, the Commodore has no provision for displaying standard characters. In order to display characters, their bit images have to be placed in the bitmap. By reading the character generator ROM (read only memory), the SETMAP2 routine is able to obtain the bit images of the characters that are needed. Simply transferring these images to the appropriate locations in the bitmap takes care of the labeling process. Another very similar program, IMAGE, is used to copy the notch and nub characters into the bitmap. The only difference being, the bit images that are read from ROM are inverted before placement in the bitmap. This causes the characters to show up in reverse video.

The final initialization task is to create data tables for use with the operational software. Tables are needed to keep track of the the present height and the present bitmap location for each of the individual bar-graphs. Additional tables are required for the present location of each of the notch and nub filters. The PRESPO routine creates these tables by writing the initial rest locations values into memory. Table 1 shows the memory

location and length for each of the tables.

B. Operational Software

After the initialization is complete, the actual process of displaying the settings of the automatic equalizer starts. Shown in Figure 13 is the flow chart of the CONTROL program. This program is responsible for determining when each of the other program modules are called, based on what operation the automatic equalizer is performing.

The first three subroutines called, BG SERV, INIT EQU and ENE SERV, draw in the bar-graphs for the very first time. The BG SERV and the ENE SERV routines will be called upon repeatedly to update their displays. The INIT EQU routine is used only once since the initial equalization display does not change.

The next operation performed by CONTROL, turns on the Commodore´s high-resolution display mode and places the start of the bitmap at $2000. Following this, the entire energy display is updated 30 separate times by the ENESERV routine. When the last energy update is completed, a check is made to determine if any of the other displays need to be updated. The appropriate service routine is called if an update is required. If not required, the energy display is updated again to start the process over.

When the automatic equalizer takes some form of corrective action, a code corresponding to the action taken is generated by its controlling software and placed at the memory location $9EFF. This code is used by the graphics display software to determine what display needs to be

TABLE I

DATA TABLE MEMORY USAGE

DATA TABLES	ADDRESSES START	END
Present Position Energy Display	$9F00	$9F3F
Present Position Band Gains (1)	$9F40	$9F7F
Present Position Band Gains (2)	$9F80	$9FBF
Present Position Init. Equ.	$9FC0	$9FFF
Energy Height	$9E00	$9E1F
Band Gains Height	$9E20	$9E3F
Init. Equ. Height	$9E40	$9E5F

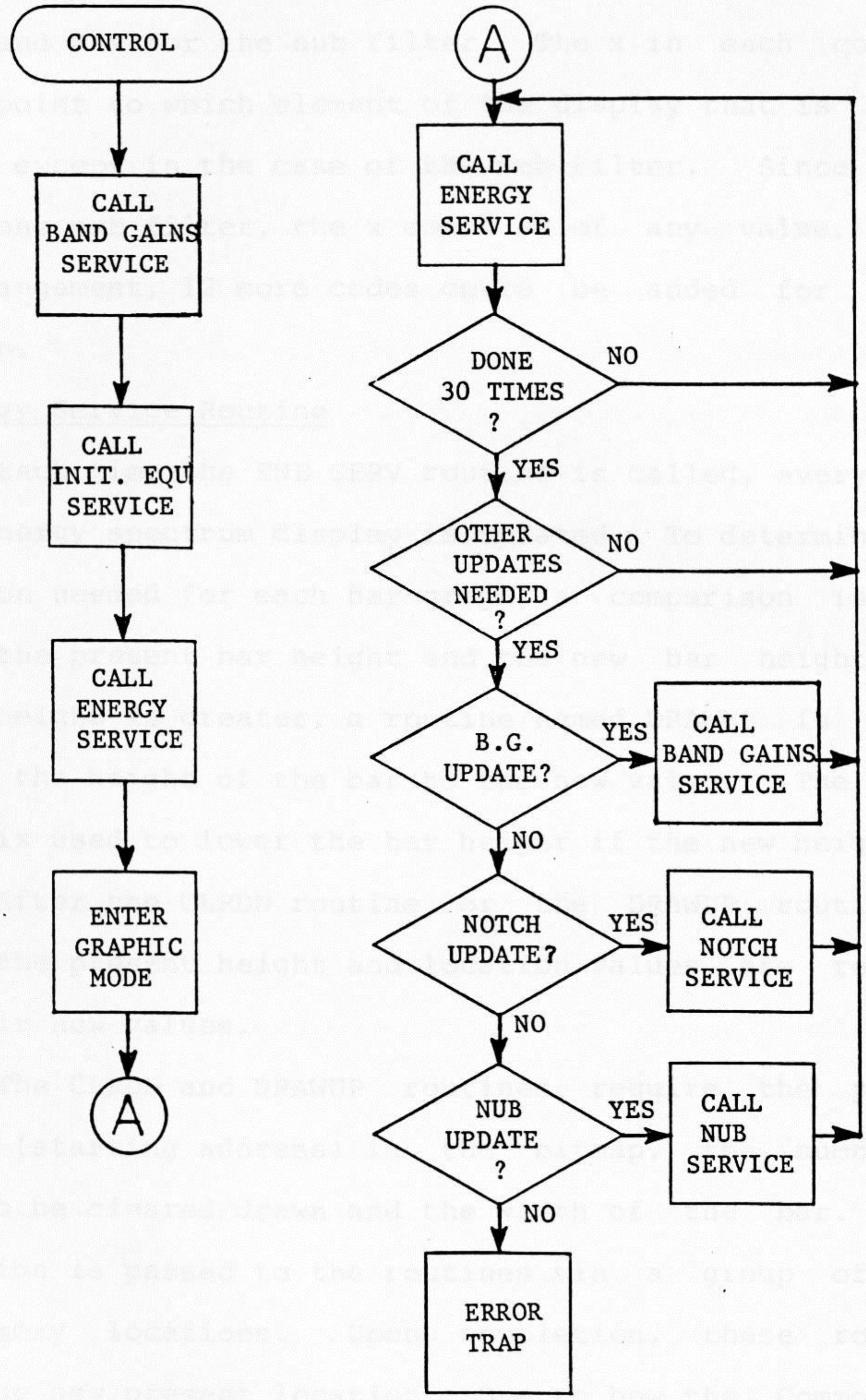

FIGURE 13 - CONTROL Program Flow Chart

updated. At present, the software implements three codes. They are: $0x and $1x for the band gains, $2x for the notch filters and $4x for the nub filter. The x in each code is used to point to which element of the display that is to be altered, except in the case of the nub filter. Since there is only one nub filter, the x could be of any value. With this arrangement, 12 more codes could be added for future expansion.

1. <u>Energy Service Routine</u>

Each time the ENE SERV routine is called, every band of the energy spectrum display is updated. To determine the correction needed for each bar-graph, a comparison is made between the present bar height and the new bar height. If the new height is greater, a routine named DRAWUP is called to raise the height of the bar to the new value. The CLRDN routine is used to lower the bar height if the new height is lower. After the CLRDN routine or the DRAWUP routine is called, the present height and location values are replaced with their new values.

The CLRDN and DRAWUP routines require the present location (starting address) in the bitmap, the number of pixels to be cleared/drawn and the width of the bar. This information is passed to the routines via a group of zero page memory locations. Upon completion, these routines return the new present location. Due to how the Commodore's screen is organized, there are difficulties involved with drawing vertical lines. For a complete discussion on what causes these difficulties and how the DRAWUP and CLRDN

routines overcome them, see Appendix A.

2. Band Gains Service Routines

There are actually two different routines for displaying the band gains. In one routine (BG1 SERV), the height of a bar-graph represents the band gain in dB. In the other routine (BG2 SERV), the height represents the difference between the present band gain and the initial band gain (initial equalization setting). If the present band gain is greater, a bar-graph will be drawn upward from the middle of the band gains display field. A downward drawn bar-graph will result when the initial band gain is larger. Figure 14 illustrates the differences in the two display formats.

The BG1 SERV routine works exactly like the ENE SERV routine except that only a single bar-graph is updated at any one time. Since the BG2 SERV routine can require bar-graphs to be drawn downward, two more clear and draw routines are needed. They are DRAWDN and CLRUP. These routines work on the same format as DRAWUP and CLRDN, and are discussed in Appendix A. The BG2 SERV routine stores the height values in two´s complement representation to keep track of negative displacements (downward draws). Using the difference form of display, it is easier to determine, at a glance, how much the present band gains are deviating from their initial settings.

3. Notch and Nub Service Routines

When the CONTROL program is alerted to a change in a notch filter´s assignment, the NCH SERV (notch service) routine is called, and the number of the filter is passed to

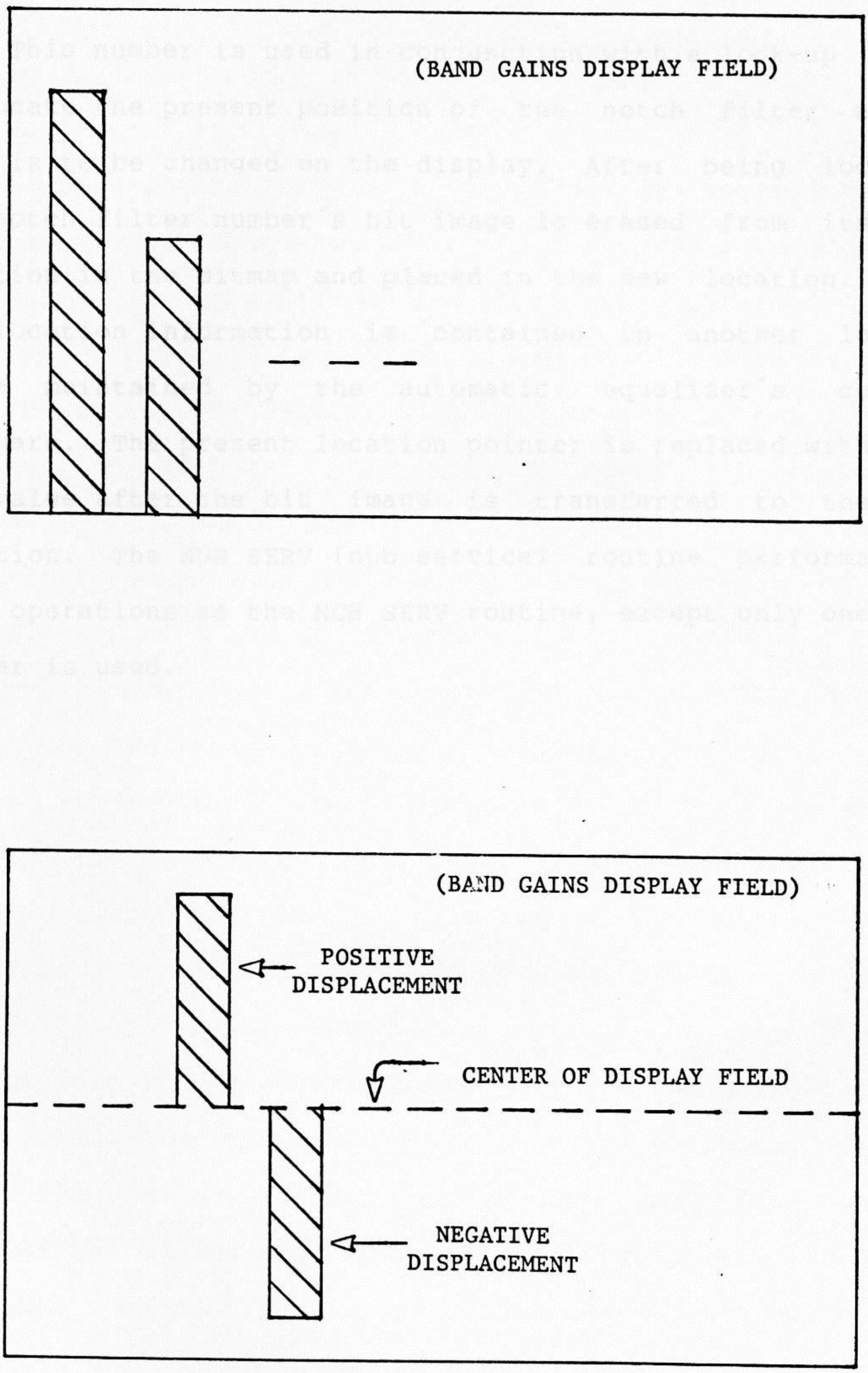

FIGURE 14 - Band Gains Display Formats

it. This number is used in conjunction with a look-up table to locate the present position of the notch filter number that is to be changed on the display. After being located, the notch filter number´s bit image is erased from its old location in the bitmap and placed in the new location. The new location information is contained in another look-up table maintained by the automatic equalizer´s control software. The present location pointer is replaced with its new value after the bit image is transferred to the new location. The NUB SERV (nub service) routine performs the same operations as the NCH SERV routine, except only one nub filter is used.

V. KEYBOARD CONTROL

With the addition of a keyboard entry routine to the graphics display software, the basis for manual control of the automatic equalizer´s functions is obtained. The Commodore´s keyboard is used to select options from a menu. When requested, the menu is displayed instead of the initial equalization display. The program named IRQ (interrupt request) takes care of both displaying the menu and detecting the keyboard entries. Both of these functions are accomplished with use of interrupts.

The Commodore´s video interface chip (VIC) has the ability to generate an interrupt based on what raster scan line it is creating. The IRQ program takes advantage of this ability, to interchange the initial equalization display and the menu. When the menu is selected to be displayed, an interrupt is set to be generated when the raster scan reaches the top of the screen. When the interrupt is generated by the VIC chip, the raster interrupt service routine turns off the graphics display mode and turns on the character display mode. The service routine then sets the next interrupt to occur at the bottom edge of the initial equalization display. When this interrupt is generated, the character display mode is turned off and the graphics display mode is turned back on. This process is shown in the flow chart of Figure 15. When the initial equalization display is selected to be displayed again, the raster interrupt service routine is

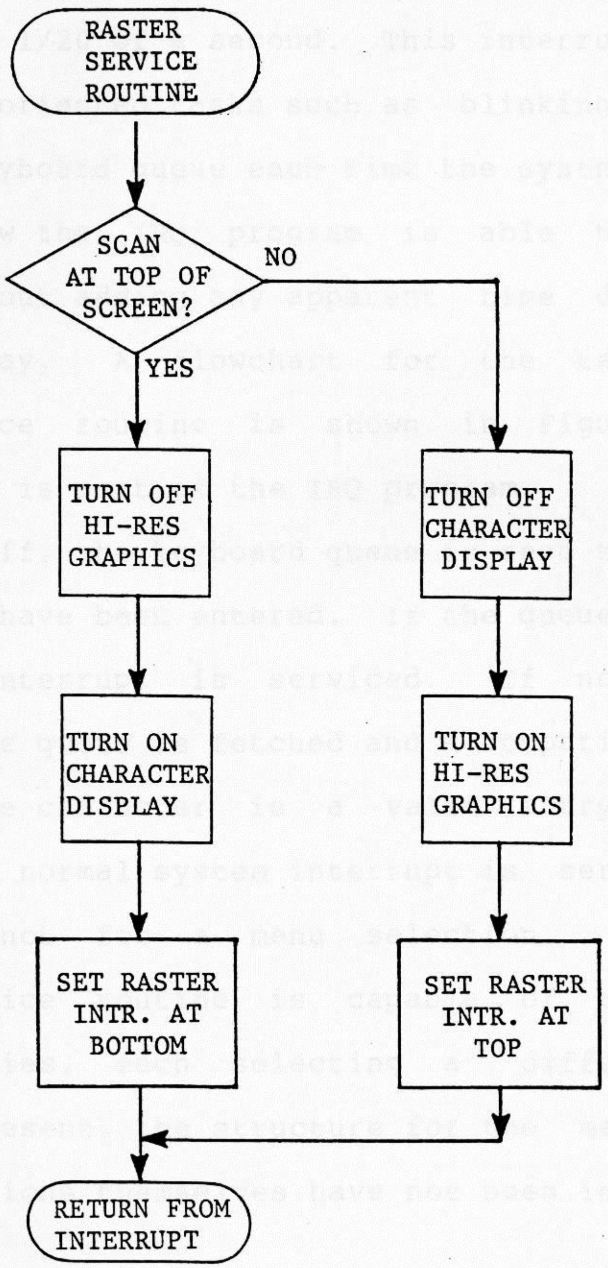

FIGURE 15 - Raster Interrupt Service Routine Flow Chart

disabled.

To take care of keyboard entries, the Commodore's system interrupt is used. The Commodore generates an interrupt every 1/20 of a second. This interrupt is used to perform system-oriented tasks such as blinking the cursor. Checking the keyboard queue each time the system interrupt is generated is how the IRQ program is able to input menu selections without adding any apparent time delays to the graphics display. A flowchart for the keyboard entry interrupt service routine is shown in Figure 16. This service routine is part of the IRQ program.

First off, the keyboard queue is read to determine if any characters have been entered. If the queue is empty, the normal system interrupt is serviced. If not empty, the character in the queue is fetched and a comparison is made to determine if the character is a valid entry for a menu selection. The normal system interrupt is serviced if the character is not for a menu selection. The keyboard interrupt service routine is capable of receiving six different entries, each selecting a different service routine. At present, the structure for the menu is merely set up; the options themselves have not been implemented.

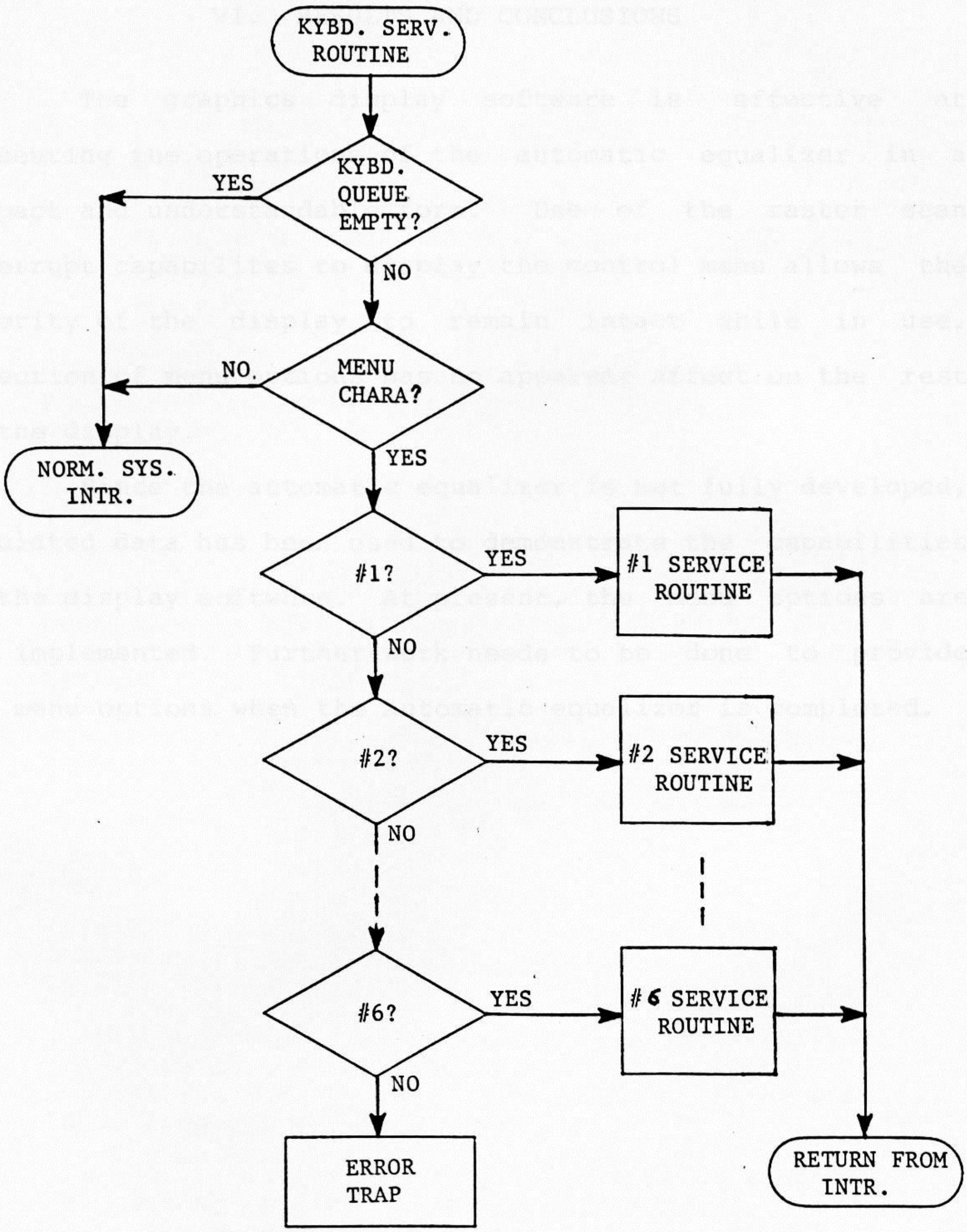

FIGURE 16 - Keyboard Interrupt Service Routine Flow Chart

VI. RESULTS AND CONCLUSIONS

The graphics display software is effective at presenting the operations of the automatic equalizer in a compact and understandable form. Use of the raster scan interrupt capabilites to display the control menu allows the majority of the display to remain intact while in use. Selection of menu options has no apparent affect on the rest of the display.

Since the automatic equalizer is not fully developed, simulated data has been used to demonstrate the capabilities of the display software. At present, the menu options are not implemented. Further work needs to be done to provide the menu options when the automatic equalizer is completed.

REFERENCES

1. Waterhouse, Richard V., "Theory of Howlback in Reverberant Rooms," Journal of Acoustic Society of America, Vol. 37 (1965), pp. 921-923.

2. Davis, Don, "A Real-Time Regenerative Response Method of Equalizing A Sound System," Journal of the Audio Engineering Society, Vol. 28 (May 1975), p. 300.

3. Ibid.

4. Bevan, William R., "A Simplified Equalization/ Analyzer," Journal of the Audio Engineering Society, Vol. 26 (March 1978), pp. 120-128.

5. Ibid.

6. Ibid.

7. Cox, J. R. and Heil, T. F., "The State-of-The-Art in Sound System Equalization," presented at the 74th Convention of the Audio Engineering Society (New York), October 8-11, 1983

8. Patronis, Eugene T., Jr., "Electronic Detection of Acoustic Feedback and Automatic Sound System Gain Control," Journal of the Audio Engineering Society, Vol. 26 (May 1978), pp. 323-325.

9. Ibid.

10. Feldman, Len, "Listening Locations Equalized Digitally," Popular Electronics, Vol. 20 (May 1982), p. 16.

11. Adams, Robert W., "An Automatic Equalizer/Analyzer," presented at the 67th Convention of the Audio Engineering Society (New York), October 31-November 3, 1980, preprint #1680 (E-3).

12. Guarino, C. R., "Audio Equalization Using Digital Processing," presented at the 63rd Convention of the Audio Engineering Society (Los Angeles), May 15-18, 1979, preprint #1486 (B-7).

13. Cox, J. Rodney, "Squealer-Killer with Real-Time Equalization," Journal of the Audio Engineering Society, Vol. 32 (March 1984), pp. 200-203

BIBLIOGRAPHY

Adams, Robert W., "An Automatic Equalizer/Analyzer," presented at the 67th Convention of the Audio Engineering Society (New York), October 31-November 3, 1980, preprint #1680 (E-3).

Angerhausen, M. and Becker, A., The Anatomy of the Commodore 64, 1st ed., Abacus Software, Missouri, 1984.

Bevan, William R., "A Simplified Equalization/ Analyzer," Journal of the Audio Engineering Society, Vol. 26 (March 1978), pp. 120-128.

Commodore 64 Programmer´s Reference Guide, 1st ed., Commodore Business Machines, Inc., 1982.

Cox, J. Rodney, "Squealer-Killer with Real-Time Equalization," Journal of the Audio Engineering Society, Vol. 32 (March 1984), pp. 200-203

Cox, J. Rodney. and Heil, T. F., "The State-of-The-Art in Sound System Equalization," presented at the 74th Convention of the Audio Engineering Society (New York), October 8-11, 1983

Davis, Don, "A Real-Time Regenerative Response Method of Equalizing A Sound System," Journal of the Audio Engineering Society, Vol. 28 (May 1975), pp. 300-302.

English, L. The Advanced Machine Language Book for the Commodore 64, 1st ed., Abacus Software, Missouri, 1984

Feldman, Len, "Listening Locations Equalized Digitally," Popular Electronics, Vol. 20 (May 1982), p. 16.

Guarino, C. R., "Audio Equalization Using Digital Processing," presented at the 63rd Convention of the Audio Engineering Society (Los Angeles), May 15-18, 1979, preprint #1486 (B-7).

Leventhal, L. A., 6502 Assembly Language Programming, 1st ed., Osborne/McGraw-Hill, California, 1979.

Patronis, Eugene T., Jr., "Electronic Detection of Acoustic Feedback and Automatic Sound System Gain Control," Journal of the Audio Engineering Society, Vol. 26 (May 1978), pp. 323-325.

Waterhouse, Richard V., "Theory of Howlback in Reverberant Rooms," Journal of Acoustic Society of America, Vol. 37 (1965).

APPENDIX A

Bitmap Display Problems

BITMAP DISPLAY PROBLEMS

The Commodore's screen is divided up into a format of 25 lines and 40 columns no matter what display mode is used. This arrangement provides for 1000 character fields. Each character field is further divided into an eight by eight matrix of pixels. The foundation for graphics storage is also this eight by eight matrix which is represented by eight bytes in the bitmap. A single byte of the bitmap yields the information for an eight pixel wide row in a character field. Figure 17 illustrates this situation.

Figure 18 shows how the bytes in the bitmap are arranged corresponding to the screen. The byte numbers represent an offset which is added to the starting address of the bitmap to locate a particular row of eight pixels. This numbering scheme presents a problem when trying to draw a vertical line up a column. For each draw upward of more than eight pixels, a boundary has to be crossed. To cross this boundary and continue upward, 313 has to subtracted from the address of the byte below the boundary. The addresses are always two byte numbers.

As an example of how the DRAWUP routine handles the situation, consider the following. The bitmap starts at \$2000. The draw is concerned with going from line one to line zero in column one. This condition is shown in Figure 19. The DRAWUP routine starts of by drawing in the first available location. For this example, the first location is

BITMAP DISPLAY PROBLEMS

The Commodore´s screen is divided up into a format of 25 lines and 40 columns no matter what display mode is used. This arrangement provides for 1000 character fields. Each character field is further divided into an eight by eight matrix of pixels. The foundation for graphics storage is also this eight by eight matrix which is represented by eight bytes in the bitmap. A single byte of the bitmap yields the information for an eight pixel wide row in a character field. Figure 17 illustrates this situation.

Figure 18 shows how the bytes in the bitmap are arranged corresponding to the screen. The byte numbers represent an offset which is added to the starting address of the bitmap to locate a particular row of eight pixels. This numbering scheme presents a problem when trying to draw a vertical line up a column. For each draw upward of more than eight pixels, a boundary has to be crossed. To cross this boundary and continue upward, 313 has to subtracted from the address of the byte below the boundary. The addresses are always two byte numbers.

As an example of how the DRAWUP routine handles the situation, consider the following. The bitmap starts at \$2000. The draw is concerned with going from line one to line zero in column one. This condition is shown in Figure 19. The DRAWUP routine starts of by drawing in the first available location. For this example, the first location is

at $214B. Next, a check is made to determine if this location lies at a boundary. The check is made by ANDing the low byte of the address with $07. This operation masks the five most significant bits. If the zero flag is set, the value $0139 (313) is subtracted from the present address. If the zero flag is not set, the address is decremented by one and a draw is made. This process continues until the destination is reached.

The CLRDN routine works in a similar manner. When the CLRDN routine encounters a boundary (from above), the value $0F is ANDed with the lobyte of the address. The lobyte is then compared to $0F. If the zero flag is not set, the value $0139 is added to the present address . If the zero flag is set, the address is incremented and another pixel group is cleared.

The two routines used with the band gains difference display, DRAWDN and CLRUP, perform the same operations as explained above. The only difference is that the draw and clear actions are reversed in direction.

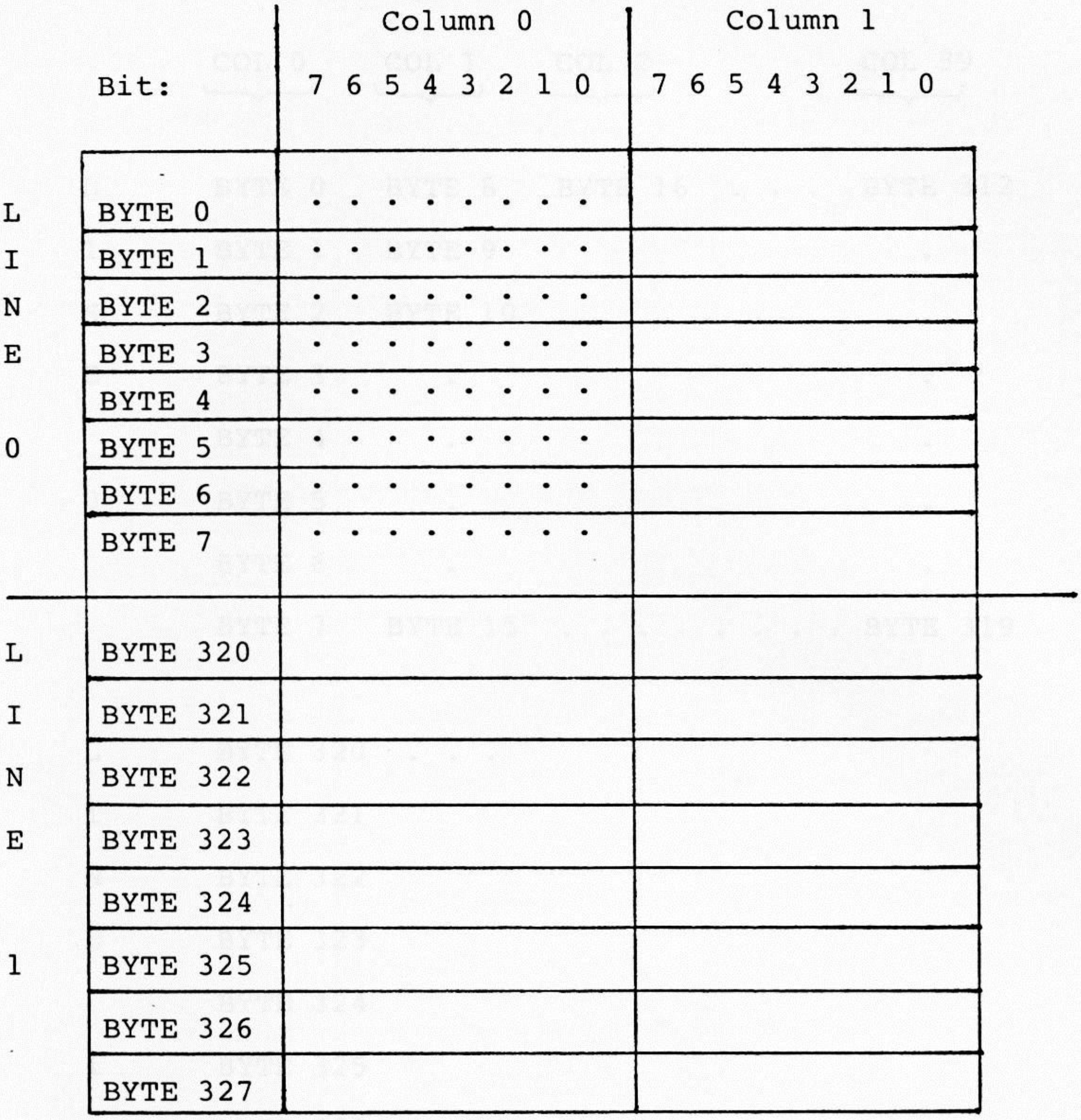

FIGURE 17 - Pixel Arrangement

	COL 0	COL 1	COL 2		COL 39
L	BYTE 0	BYTE 8	BYTE 16	. . .	BYTE 312
I	BYTE 1	BYTE 9			.
N	BYTE 2	BYTE 10			.
E	BYTE 3	.			.
	BYTE 4	.			.
0	BYTE 5	.			.
	BYTE 6	.			.
	BYTE 7	BYTE 15	. . . .		BYTE 319
L	BYTE 320	. . .			
I	BYTE 321				
N	BYTE 322				
E	BYTE 323				
	BYTE 324				
1	BYTE 325				
	BYTE 326				
	BYTE 327				

FIGURE 18 - Bitmap Numbering

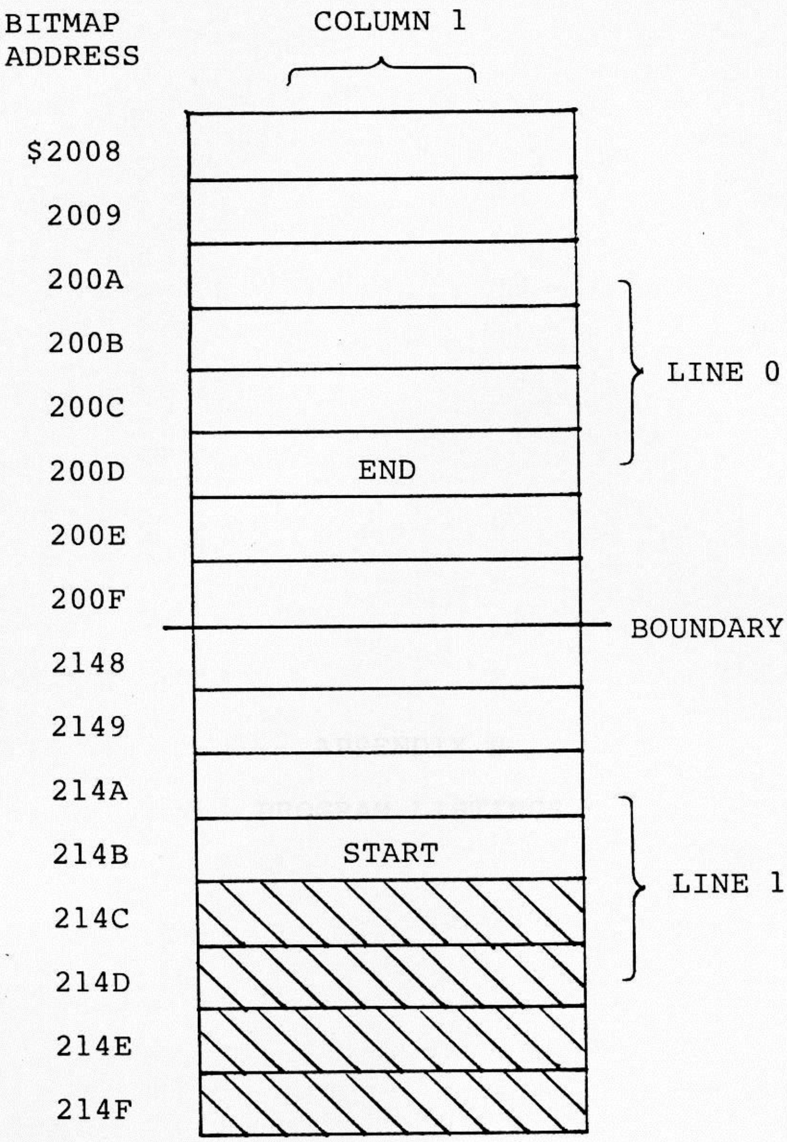

FIGURE 19 - DRAWUP Example

APPENDIX B

PROGRAM LISTINGS

```
;NAME: TRIAL CONTROL
;PURPOSE: TO CONTROL PROGRAM FLOW
;
;********************************
;
;ADDRESSES OF THE SERVICE ROUTINES
;
       BGSERV = $9800
       ENESER = $9500
       NCHSER = $9648
       NUBSER = $9742
       INISER = $9570
       MAPCLR = $9000
       COLOR1 = $9020
       COLOR2 = $9040
       MAP1   = $9100
       MAP2   = $924C
       IMAGE  = $9090
       PRESPO = $9312
       IRQENA = $9D00
;
;NOTCH LOCATORS IN COX MAP
;
       COXNCH = $8F61
;
;NUB LOCATOR IN COX MAP
;
       NUBCOX = $8F60
;
;ENERGY HEIGHT IN COX MAP
;
       COXENE = $8F00
;
;BG HEIGHT IN COX MAP
;
       COXBG  = $8F20
;
;INI EQU HEIGHT IN COX MAP
;
       COXEQU = $8F40
;
;CONTROL LOCATION FOR SERVICE ROUTINES
;
       CONTRL = $9EFE
;
;LOCATION OF THE INI EQU DATA
;
       INIDAT = $8F65
;
;COUNTER LOCATION FOR ENERGY DISPLAY
;
       ENECNT = $9E7F
;
       * = $9A00
```

```
       JSR MAPCLR    ;CONFIGURE
       JSR COLOR1    ;
       JSR COLOR2    ; BIT
       JSR MAP1      ;
       JSR MAP2      ;  MAP
       JSR IMAGE
       JSR PRESPO
       JSR IRQENA
       LDA #$00      ;LOBYTES FOR
       STA $A3       ; ENEDATA
       STA $A5       ;SERVICE DATA
       LDA #$60      ;HIBYTE
       STA $A4       ; ENEDATA
       LDA #$7D      ;HIBYTE
       STA $A6       ; SERVICE DATA
       LDX #$00
       LDY #$00
ENEINT LDA ($A3),Y   ;GET INITIAL
       STA COXENE,X  ; ENERGY DATA
       INC $A3
       INX
       CPX #$20      ;32 TIMES
       BNE ENEINT
       LDX #$00      ;GET INITIAL
BGINT  LDA INIDAT,X  ; BG AND EQU
       STA COXBG,X   ;  DATA
       STA COXEQU,X
       INX
       CPX #$20      ;32 TIMES
       BNE BGINT
       LDX #$00      ;SET NOTCH
       LDA #$FF      ; AND NUB
REST   STA NUBCOX,X  ;  LOCATIONS
       INX           ;   TO REST
       CPX #$05
       BNE REST
;
;DONE WITH INITIALIZATION
;
       JSR INISER
       JSR ENESER
       LDX #$00
       STX CONTRL
BGLOOP JSR BGSERV    ;CALL
       INC CONTRL    ; BGSERV
       LDX CONTRL    ;  32
       CPX #$20      ;   TIMES
       BNE BGLOOP
;
;DONE WITH START UP DISPLAY
;
       LDA $D018     ;PLACE
       ORA #$08      ; BIT-MAP
       STA $D018     ;  AT $2000
```

```
BGPRP   LDA $D011     ;ENTER
        ORA #$20      ; BIT-MAP
        STA $D011     ;  MODE
        LDA #$06      ;BLUE
        STA $D020     ; BORDER
        LDA #$00      ;CLEAR
        STA ENECNT    ; ENECOUNTER
DELAY   LDX #$DB      ;TIME
XLOOP   LDY #$00      ; DELAY
YLOOP   NOP
        INY
        BNE YLOOP
        INX
        BNE XLOOP     ;DELAY DONE
        LDX #$00      ;GET
        LDY #$00      ; NEXT
NEXDIS  LDA ($A3),Y   ;  ENERGY
        STA COXENE,X  ;   DATA
        INX
        INC $A3       ;CHECK
        BNE XCOUNT    ;
        LDA $A4       ; FOR
        CMP #$7C      ;
        BNE NWRAP1    ;  END
        LDA #$60      ;
        STA $A4       ;   OF
        JMP XCOUNT    ;
NWRAP1  INC $A4       ;    DATA
XCOUNT  CPX #$20
        BNE NEXDIS
        JSR ENESER
        INC ENECNT    ;DO
        LDA ENECNT    ; 30
        CMP #$1E      ;  TIMES
        BNE DELAY
        LDY #$00      ;CLEAR
        STY ENECNT    ; ENE COUNTER
SERV    LDA ($A5),Y   ;NEXT
        TAX           ; SERVICE
        JSR INCRE     ;  COMMAND
        ASL A         ;CHECK FOR
        BMI NUBPRP    ; NUB CODE
        ASL A         ;CHECK FOR
        BMI NCHPRP    ; NOTCH CODE
        ASL A         ;CHECK
        BPL LOWER     ; FOR
        TXA           ;  UPPER
        AND #$0F      ;   OR
        ORA #$10      ;    LOWER
        STA CONTRL    ;     BG
        JMP BGPRP
LOWER   TXA
        AND #$0F
        STA CONTRL
```

```
BGPRP  TAX              ;GET
       LDA ($A5),Y      ; NEW
       STA COXBG,X      ;  BG
       JSR INCRE        ;   DATA
       JSR BGSERV
       JMP DELAY2
NUBPRP TXA
       AND #$0F
       STA CONTRL
       LDA ($A5),Y      ;SERVICE
       JSR INCRE        ; DATA
       STA NUBCOX
       JSR NUBSER
       JMP DELAY2
NCHPRP TXA              ;NOTCH
       AND #$0F         ; SERVICE #
       STA CONTRL
       TAX
       LDA ($A5),Y
       STA COXNCH,X
       JSR INCRE
       JSR NCHSER
DELAY2 LDX #$EA         ;TIME
XLOOP2 LDY #$00         ; DELAY (2)
YLOOP2 NOP
       INY
       BNE YLOOP2
       INX
       BNE XLOOP2
       JMP NEXDIS       ;DELAY DONE
INCRE  INC $A5          ;INCREMENT
       BNE DONINC       ; DATA
       LDA $A6          ;  POINTER
       CMP #$7E         ;   AND
       BNE NWRAP2       ;    CHECK
       LDA #$7D         ;     FOR
       STA $A6          ;      END
       JMP DONINC
NWRAP2 INC $A6
DONINC RTS
```

```
;NAME: IRQ
;
;
;**********************************
;
        IRQVCT = $0314
        IRQOLD = $EA31
        RASTER = $D012
        GETIN  = $FFE4
        IRR    = $D019
        IMR    = $D01A
        UP     = $FB
        DN     = $FD
        FLAG   = $9D2F
;
;INITIALIZE
;
        * = $9D00
        SEI            ;DISABLE INTR
        LDA #$30       ;NEW
        STA IRQVCT     ; INTERRUPT
        LDA #$9D       ;  SERVICE
        STA IRQVCT+1   ;   LOCATION
        LDA #$1E       ;SET FIRST
        STA RASTER     ; RASTER LINE
        LDA RASTER-1   ;CLEAR
        AND #$7F       ; MS BIT
        STA RASTER-1
        LDA #$81       ;SELECT
        STA IMR        ; RASTER IRQ
        LDA #$FF       ;SET DISPLAY
        STA FLAG       ; STATE FLAG
        CLI            ;  ENABLE INTR
        RTS
;
;INTERRUPT ROUTINE
;
        * = $9D30
        LDA IRR        ;GET INTR
        STA IRR        ; REGISTER
        BMI RASSER     ;MSB SET ?
        LDA $DC0D      ;CLEAR CIA
        CLI            ; TIMER INTR
;
;GET KEYBOARD ENTRY
;
        LDA $C5        ;
        CMP #$04       ;F1 KEY
        BNE CIA        ; DEPRESSED ?
        LDA FLAG       ;CHECK STATE
        BEQ IRQIN      ; OF DISPLAY
;
;NORMAL SCREEN ENABLE
;
```

```
        SEI           ;DISABLE INTR
        LDA IRR       ;CLEAR POSSIBLE
        STA IRR       ; RASTER INTR
        LDA #$80      ;CLEAR RASTER
        STA IMR       ; IRQ MASK
        LDA #$18      ;PLACE SCREEN
        STA $D018     ; AT $0400
        LDA #$04      ;INFORM SCREEN
        STA $0288     ; EDITOR
        LDA #$00      ;CLEAR
        STA FLAG      ; FLAG
        JMP IRQOLD
;
;SPLIT SCREEN ENABLE
;
IRQIN   LDA #$1E      ;SET FIRST
        STA RASTER    ; RASTER LINE
        LDA #$81      ;SELECT
        STA IMR       ; RASTER IRQ
        LDA #$FF      ;SET
        STA FLAG      ; FLAG
;
;NORMAL INTERRUPT (CIA TIMER)
;
CIA     JMP IRQOLD    ;OLD ROUTINE
;
;NEW ROUTINE
;
RASSER LDA RASTER     ;POSITION
        CMP #$62      ; LOW ON
        BEQ LOW       ;  SCREEN
        CMP #$1E      ;CHECK FOR
        BNE RESET     ; HIGH INTR
        LDA $D011     ;TURN
        AND #$DF      ; GRAPHICS
        STA $D011     ;  OFF
        LDA #$F4      ;PLACE SCREEN
        STA $D018     ; AT $3C00
        LDA #$3C      ;INFORM SCREEN
        STA $0288     ;EDITOR
        LDA #$62      ;NEXT INTR
        STA RASTER    ; LOW
RESET   CLI
        JMP $FEBC     ;RESET STACK
LOW     LDA $D011     ;TURN
        ORA #$20      ; GRAPHICS
        STA $D011     ;  ON
        LDA #$18      ;PLACE SCREEN
        STA $D018     ; AT 0400
        LDA #$04      ;INFORM SCREEN
        STA $0288     ; EDITOR
        LDA #$1E      ;NEXT INTR
        STA RASTER    ; HIGH
        CLI
```

```
;NAME: INISERV
;PURPOSJMP $FEBCATE THE INITIAL
;          EQULIZATION DISPLAY
;
;*******************************
;
;ALL ZERO PAGE EQUATES ARE MEMORY
;LOCATIONS USED BY THE PLOT ROUTINE
;
        PLOTLO = $FB
        PLOTHI = $FC
        SHAPE  = $FD
        HEIGHT = $FE
;
;ADDRESS OF PLOT ROUTINE
;
        CLRDN  = $9430
        DRAWUP = $9400
;
; NEW HEIGHT IN COX MAP
;
        EQUC   = $8F40
;
; PRESENT POSITION OF INIT EQU DISPLAYS
;
        PRESEQ = $9FC0
;
; PRESENT HEIGHT IN STORTZ MAP
;
        EQUS   = $9E40
;
        XREG   = $9E60 ;TEMP REG
        YREG   = $9E61 ; STORAGE
;
       * = $9570
       LDX #$00       ;CLEARING
       STX XREG       ; COUNTERS
       LDY #$00       ;  USED
       STY YREG       ;
       LDA #$3C       ;DRAW SHAPE
       STA SHAPE      ; FOR PLOT
FIRST  LDA PRESEQ,X   ;GET
       STA PLOTLO     ; PRESENT
       INX            ;  POSITION
       LDA PRESEQ,X   ;   FOR
       STA PLOTHI     ;    PLOT
       LDA EQUC,Y     ;CALCULATE
       PHA            ; HEIGHT
       SEC            ;  DIFFERENCE
       SBC EQUS,Y
       BEQ INCR       ;CHECK IF EQUAL
       BCS DRAW       ;BORROW CLEAR
       EOR #$FF       ;TAKE
       CLC            ; TWO'S
```

```
;NAME: INISERV
;PURPOSE: TO UPDATE THE INITIAL
;         EQULIZATION DISPLAY
;
;*********************************
;
;ALL ZERO PAGE EQUATES ARE MEMORY
;LOCATIONS USED BY THE PLOT ROUTINE
;
        PLOTLO = $FB
        PLOTHI = $FC
        SHAPE  = $FD
        HEIGHT = $FE
;
;ADDRESS OF PLOT ROUTINE
;
        CLRDN  = $9430
        DRAWUP = $9400
;
; NEW HEIGHT IN COX MAP
;
        EQUC   = $8F40
;
; PRESENT POSITION OF INIT EQU DISPLAYS
;
        PRESEQ = $9FC0
;
; PRESENT HEIGHT IN STORTZ MAP
;
        EQUS   = $9E40
;
        XREG   = $9E60 ;TEMP REG
        YREG   = $9E61 ; STORAGE
;
       * = $9570
       LDX #$00      ;CLEARING
       STX XREG      ; COUNTERS
       LDY #$00      ;  USED
       STY YREG      ;
       LDA #$3C      ;DRAW SHAPE
       STA SHAPE     ; FOR PLOT
FIRST  LDA PRESEQ,X  ;GET
       STA PLOTLO    ; PRESENT
       INX           ;  POSITION
       LDA PRESEQ,X  ;   FOR
       STA PLOTHI    ;    PLOT
       LDA EQUC,Y    ;CALCULATE
       PHA           ; HEIGHT
       SEC           ;  DIFFERENCE
       SBC EQUS,Y
       BEQ INCR      ;CHECK IF EQUAL
       BCS DRAW      ;BORROW CLEAR
       EOR #$FF      ;TAKE
       CLC           ; TWO´S
```

```
        ADC #$01      ; COMP
        STA HEIGHT    ;PLOT
        PLA           ;UPDATE
        STA EQUS,Y    ; HEIGHT
        JSR CLRDN
        JMP UPDATE
DRAW    STA HEIGHT    ;DRAW
        PLA           ;UPDATE
        STA EQUS,Y    ; HEIGHT
        JSR DRAWUP
UPDATE  LDX XREG      ;UPDATE
        LDA PLOTLO    ;
        STA PRESEQ,X  ; PRESENT
        INX           ;
        LDA PLOTHI    ;  POSITION
        STA PRESEQ,X  ;
        INX
        STX XREG
        LDY YREG
        INY
        STY YREG
        CPY #$20      ;CHECK FOR
        BNE FIRST     ; 32 TIMES
DONE    RTS           ;ALL DONE
INCR    PLA           ;RESET STACK
        INY           ;CHECK
        CPY #$20      ; IF
        BEQ DONE      ;  DONE
        STY YREG      ;UPDATE
        INX           ; COUNTERS
        STX XREG
        JMP FIRST     ;DO AGAIN
```

```
;NAME: ENESERV
;PURPOSE: TO UPDATE THE ENERGY
;         DISPLAY
;
;*********************************
;
;ALL ZERO PAGE EQUATES ARE MEMORY
;LOCATIONS USED BY THE PLOT ROUTINE
;
        PLOTLO = $FB
        PLOTHI = $FC
        SHAPE  = $FD
        HEIGHT = $FE
;
;ADDRESS OF PLOT ROUTINE
;
        CLRDN  = $9430
        DRAWUP = $9400
;
; NEW HEIGHT IN COX MAP
;
        COXENE = $8F00
;
; PRESENT POSITION OF ENERGY DISPLAYS
;
        PRESEN = $9F00
;
; PRESENT HEIGHT IN STORTZ MAP
;
        ENEHEI = $9E00
        XREG   = $9E60 ;TEMP REG
        YREG   = $9E61 ; STORAGE
;
       * = $9500
       LDX #$00      ;CLEARING
       STX XREG      ; COUNTERS
       LDY #$00      ;  USED
       STY YREG
       LDA #$18      ;DRAW SHAPE
       STA SHAPE     ; FOR PLOT
FIRST  LDA PRESEN,X ;GET
       STA PLOTLO    ; PRESENT
       INX           ;  POSITION
       LDA PRESEN,X ;    FOR
       STA PLOTHI    ;     PLOT
       LDA COXENE,Y ;CALCULATE
       PHA           ; HEIGHT
       SEC           ;  DIFFERENCE
       SBC ENEHEI,Y
       BEQ INCR      ;CHECK IF EQUAL
       BCS DRAW      ;BORROW CLEAR
       EOR #$FF      ;TAKE
       CLC           ; TWO´S
       ADC #$01      ;  COMP
```

```
       STA HEIGHT ;PLOT
       PLA        ;UPDATE
       STA ENEHEI,Y ; HEIGHT
       JSR CLRDN
       JMP UPDATE
DRAW   STA HEIGHT ;DRAW
       PLA        ;UPDATE
       STA ENEHEI,Y ; HEIGHT
       JSR DRAWUP
UPDATE LDX XREG   ;UPDATE
       LDA PLOTLO ;
       STA PRESEN,X ; PRESENT
       INX        ;
       LDA PLOTHI ;  POSITION
       STA PRESEN,X ;
       INX
       STX XREG
       LDY YREG
       INY
       STY YREG
       CPY #$20   ;CHECK FOR
       BNE FIRST  ; 32 TIMES
DONE   RTS        ;ALL DONE
INCR   PLA        ;RESET STACK
       INY        ;CHECK
       CPY #$20   ; IF
       BEQ DONE   ;  DONE
       STY YREG   ;UPDATE
       INX        ; COUNTERS
       STX XREG
       JMP FIRST  ;DO AGAIN
```

```
;NAME: BG2SERV
;PURPOSE: TO UPDATE THE BAND GAIN
;        DISPLAY  (2)
;
;********************************
;
;ALL ZERO PAGE EQUATES ARE MEMORY
;LOCATIONS USED BY PLOT ROUTINE
;
        PLOTLO = $FB ;LOBYTE
        PLOTHI = $FC ;HIBYTE
        SHAPE  = $FD ;DRAW SHAPE
        HEIGHT = $FE ;HEIGHT DATA
;
;ADDRESSES OF PLOT ROUTINES
;
        CLRDN  = $9430
        DRAWUP = $9400
        CLRUP  = $9460
        DRAWDN = $9490
;
;PRESENT BG HEIGHT IN STORTZ MAP
;
        BGHEIS = $9E20
;
;NEW BG HEIGHT IN COX MAP
;
        BGHEIC = $8F20
;
;CONTROL MEMEORY LOCATION
;
        CONTRL = $8FFF
;
;INITIAL EQ HEIGHT IN STORTZ MAP
;
        INIHEI = $9E40
;
;PRESENT BG POSITION IN STORTZ MAP
;
        PRESBG = $9F80
;
;TEMPORARY REGISTER STORAGE AREA
;
        XREG   = $9E60
;
        * = $9900
        LDA #$3C       ;DRAW SHAPE
        STA SHAPE
        LDA CONTRL     ;BG NUMBER
        AND #$0F
        TAX
        STX XREG
        ASL A          ;MULT BY 2
        PHA
```

```
        TAY
        LDA PRESBG,Y ;PRESENT
        STA PLOTLO   ; POSITION
        INY
        LDA PRESBG,Y
        STA PLOTHI
        LDA BGHEIC,X ;NEW HEIGHT
        SEC
        SBC INIHEI,X ;INITIAL HEIGHT
        BCC BELOW    ;BORROW SET
;
;NEW HEIGHT ABOVE THE AXIS
;
        CMP #$18     ;CHECK FOR
        BCC LESMAX   ; MAX HEIGHT
        LDA #$18
LESMAX  PHA          ;DIFF HEIGHT
        LDA BGHEIS,X ;PRES HEIGHT
        ASL A        ;CHECK FOR
        BCC ABOVE    ; MS BIT SET
;
;PRESENT HEIGHT BELOW AXIS
;
        LSR A
        AND #$7F     ;CLEAR MS BIT
        CLC          ;INCR
        ADC #$01     ; PRES HEIGHT
        STA HEIGHT
        JSR CLRUP
        LDX XREG
        PLA          ;DIFF HEIGHT
        STA BGHEIS,X ;UPDATE HEIGHT
        BNE GOON
        JMP UPDATE
GOON    STA HEIGHT   ;PLOT
        JSR DRAWUP
        JMP UPDATE
;
;PRESENT  HEIGHT ABOVE AXIS
;
ABOVE   PLA          ;DIFF HEIGHT
        TAY
        SEC          ;CLR BORROW
        SBC BGHEIS,X ;PRES HEIGHT
        BNE CONTIN   ;TOO BIG FOR
        JMP EQUAL    ; RELATIVE
CONTIN  BCS DUP      ;  BRANCH
        EOR #$FF     ;TAKE
        CLC          ; TWO'S
        ADC #$01     ;  COMP
        STA HEIGHT
        TYA          ;UPDATE
        STA BGHEIS,X ; HEIGHT
        JSR CLRDN
```

```
        JMP UPDATE
DUP     STA HEIGHT
        TYA            ;UPDATE
        STA BGHEIS,X ; HEIGHT
        JSR DRAWUP
        JMP UPDATE
;
;NEW HEIGHT BELOW AXIS
;
BELOW   EOR #$FF       ;TAKE
        CLC            ; TWO´S
        ADC #$01       ;  COMP
        CMP #$18       ;CHECK FOR
        BCC LTMAX      ; MAX HEIGHT
        LDA #$18
LTMAX   PHA            ;DIFF HEIGHT
        LDA BGHEIS,X ;PRES HEIGHT
        ASL A          ;CHECK
        BCS BELOW1     ; MSB
;
;PRESENT HEIGHT ABOVE AXIS
;
        LSR A
        CLC            ;INCR
        ADC #$01       ; PRES HEIGHT
        STA HEIGHT     ;PLOT
        JSR CLRDN
        PLA            ;DIFF HEIGHT
        STA HEIGHT
        ORA #$80       ;SET MS BIT
        LDX XREG
        STA BGHEIS,X ;UPDATE HEIGHT
        JSR DRAWDN
        JMP UPDATE
;
;PRESENT HEIGHT BELOW AXIS
;
BELOW1  LSR A
        AND #$7F       ;CLEAR MS BIT
        STA HEIGHT     ;TEMP STORAGE
        PLA            ;DIFF HEIGHT
        TAY
        SEC
        SBC HEIGHT
        BEQ EQUAL
        BCS DDOWN
        EOR #$FF       ;TAKE
        CLC            ; TWO´S
        ADC #$01       ;  COMP
        STA HEIGHT     ;PLOT
        TYA
        ORA #$80       ;SET MS BIT
        STA BGHEIS,X ;UPDATE HEIGHT
        JSR CLRUP
```

```
        JMP UPDATE
DDOWN   STA HEIGHT   ;PLOT
        TYA
        ORA #$80     ;SET MS BIT
        STA BGHEIS,X ;UPDATE HEIGHT
        JSR DRAWDN
UPDATE  PLA
        TAY
        LDA PLOTLO   ;UPDATE
        STA PRESBG,Y ; PRESENT
        INY          ;  POSITON
        LDA PLOTHI
        STA PRESBG,Y
DONE    RTS          ;ALL DONE
EQUAL   PLA          ;SAME HEIGHT
        RTS
```

```
;NAME: BG1SERV
;PURPOSE: TO UPDATE THE BAND GAIN
;         DISPLAY (1)
;
;*********************************
;
;ALL ZERO PAGE EQUATES ARE MEMORY
;LOCATIONS USED BY PLOT ROUTINE
;
        PLOTLO = $FB ;LOBYTE POS
        PLOTHI = $FC ;HIBYTE POS
        SHAPE  = $FD ;DRAW SHAPE
        HEIGHT = $FE ;HEIGHT DATA
;
;ADDRESSES OF PLOT ROUTINE
;
        DRAWUP = $9400
        CLRDN  = $9430
;
;PRESENT HEIGHT IN STORTZ MAP
;
        BGHEIS = $9E20
;
;NEW HEIGHT IN COX MAP
;
        BGHEIC = $8F20
;
;CONTROL LOCATION
;
        CONTRL = $8FFF
;
;PRESENT POSITION OF BG DISPLAYS
;
        PRESBG = $9F40
;
           * = $9800
        LDA #$3C      ;DRAW SHAPE
        STA SHAPE
        LDA CONTRL    ;BAND GAIN
        AND #$0F      ; NUMBER
        TAY
        ASL A         ;MULT BY 2
        PHA           ;LATER USE
        TAX
        LDA PRESBG,X ;PRESENT
        STA PLOTLO    ; POSITION
        INX           ;  FOR
        LDA PRESBG,X ;   PLOT
        STA PLOTHI
        LDA BGHEIC,Y ;NEW HEIGHT
        TAX           ;CALCULATE
        SEC           ; HEIGHT
        SBC BGHEIS,Y ;  DIFFERENCE
        BEQ EQUAL     ;SAME VALUE
```

```
        BCS DRAW       ;BORROW CLEAR
        EOR #$FF       ;TAKE
        CLC            ; TWO´S
        ADC #$01       ;  COMP
        STA HEIGHT     ;PLOT
        TXA            ;UPDATE PRES
        STA BGHEIS,Y   ; HEIGHT
        JSR CLRDN
        JMP UPDATE
DRAW    STA HEIGHT     ;PLOT
        TXA            ;UPDATE PRES
        STA BGHEIS,Y   ; HEIGHT
        JSR DRAWUP
UPDATE  PLA
        TAX
        LDA PLOTLO     ;UPDATE
        STA PRESBG,X   ; PRESENT
        INX            ;  POSITION
        LDA PLOTHI
        STA PRESBG,X
DONE    RTS            ;ALL DONE
EQUAL   PLA
        RTS
```

```
;NAME: NCHSERV
;PURPOSE: TO UPDATE THE NOTCH DISPLAY
;
;***********************************
;
; NOTCH LOCATORS IN STORTZ MAP
;
       NOTCHS = $9EB1
;
; NOTCH LOCATORS IN COX MAP
;
       NOTCHC = $8F61
;
; NOTCH IMAGE LOCATIONS STORTZ MAP
;
       IMAGE  = $9E88
;
; CONTROL LOCATION
;
       CONTRL = $8FFF
;
         * = $9600
;
; ADDRESSES FOR THE 32 LOCATIONS
;
PLACE  .WORD $3180,$3188,$3190
       .WORD $3198,$31A0,$31A8
       .WORD $31B0,$31B8,$31C0
       .WORD $31C8,$31D0,$31D8
       .WORD $31E0,$31E8,$31F0
       .WORD $31F8,$3200,$3208
       .WORD $3210,$3218,$3220
       .WORD $3228,$3230,$3238
       .WORD $3240,$3248,$3250
       .WORD $3258,$3260,$3268
       .WORD $3270,$3278
;
;ADDRESSES FOR THE REST LOCATIONS
;
REST   .WORD $3288,$3298
       .WORD $32A8,$32B8
;
       LDA CONTRL   ;GET NOTCH
       AND #$0F     ; NUMBER
       PHA
       TAY
       TAX
       LDA NOTCHS,Y ;PRES LOCATION
       CMP #$FF     ;CHECK FOR
       BNE NREST    ; REST
       TYA
       ASL A        ;MULT BY 2
       TAY
       LDA REST,Y   ;LOBYTE OF
```

```
        STA $FB     ; PRES POS
        INY
        LDA REST,Y  ;HIBYTE
        STA $FC
        JMP NEWLOC
NREST   ASL A       ;MULT BY 2
        TAY
        LDA PLACE,Y ;LOBYTE OF
        STA $FB     ; PRES POS
        INY
        LDA PLACE,Y ;HIBYTE
        STA $FC
NEWLOC  LDA NOTCHC,X ;NEW LOCATION
        PHA
        CMP #$FF    ;CHECK FOR
        BNE NREST1  ; REST
        TXA
        ASL A       ;MULT BY 2
        TAX
        LDA REST,X  ;LOBYTE OF
        STA $FD     ; NEW LOCATION
        INX
        LDA REST,X ;HIBYTE
        STA $FE
        JMP UPDATE
NREST1  ASL A       ;MULT BY 2
        TAX
        LDA PLACE,X
        STA $FD
        INX
        LDA PLACE,X
        STA $FE
UPDATE  PLA        ;NEW
        TAY        ; LOCATION #
        PLA        ;NOTCH
        TAX        ; NUMBER
        TYA
        STA NOTCHS,X ;UPDATE LOC
        TXA
        LDY #$00
        ASL A       ;MULTIPLY
        ASL A       ; BY
        ASL A       ;  8
        TAX
AGAIN   LDA #$00    ;CLEAR
        STA ($FB),Y ; BYTE
        LDA IMAGE,X ;FILL
        STA ($FD),Y ; IN
        INX
        INY
        CPY #$08
        BNE AGAIN
        RTS         ; ALL DONE
```

```
;NAME: NUBSERV
;PURPOSE: TO UPDATE THE NUB DISPLAY
;
;*********************************
;
        NUBS   = $9EB0
        NUBC   = $8F60
        IMAGE  = $9E80
        CONTRL = $8FFF
;
             * = $9700
;
; ADDRESS FOR THE NUB´S REST LOCATION
;
REST    .WORD $3DD8
;
; ADDRESSES FOR THE 32 LOCATIONS
;
PLACE   .WORD $3CC0,$3CC8,$3CD0
        .WORD $3CD8,$3CE0,$3CE8
        .WORD $3CF0,$3CF8,$3D00
        .WORD $3D08,$3D10,$3D18
        .WORD $3D20,$3D28,$3D30
        .WORD $3D38,$3D40,$3D48
        .WORD $3D50,$3D58,$3D60
        .WORD $3D68,$3D70,$3D78
        .WORD $3D80,$3D88,$3D90
        .WORD $3D98,$3DA0,$3DA8
        .WORD $3DB0,$3DB8
;
        LDA NUBS      ;PRES LOCATION
        CMP #$FF      ;CHECK FOR
        BNE NREST     ; REST
        LDY #$00
        LDA REST,Y    ;LOBYTE OF
        STA $FB       ; PRES POS
        INY
        LDA REST,Y    ;HIBYTE
        STA $FC
        JMP NEWLOC
NREST   ASL A         ;MULT BY 2
        TAY
        LDA PLACE,Y   ;LOBYTE OF
        STA $FB       ; PRES POS
        INY
        LDA PLACE,Y   ;HIBYTE
        STA $FC
NEWLOC  LDA NUBC      ;NEW LOCATION
        CMP #$FF      ;CHECK FOR
        BNE NREST1    ; REST
        LDY #$00
        LDA REST,Y    ;LOBYTE OF
        STA $FD       ; NEW POS
        INY
```

```
        LDA REST,Y   ;HIBYTE
        STA $FE
        JMP UPDATE
NREST1  ASL A        ;MULT BY 2
        TAY
        LDA PLACE,Y  ;LOBYTE OF
        STA $FD      ; NEW POS
        INY
        LDA PLACE,Y  ;HIBYTE
        STA $FE
UPDATE  LDA NUBC     ;UPDATE STORTZ
        STA NUBS     ; MAP
        LDY #$00
AGAIN   LDA #$00     ;CLEAR BYTE
        STA ($FB),Y  ; BYTE
        LDA IMAGE,Y  ;FILL IN
        STA ($FD),Y  ; BYTE
        INY
        CPY #$08
        BNE AGAIN
        RTS          ;ALL DONE
```

```
; NAME: IMAGE
; PURPOSE: TO GET IMAGES FOR NOTCH
; & NUB CHARACTERS AND PLACE IN
; STORAGE FOR PROGRAM USE, ALSO TO
; PUT THE IMAGES AT THEIR INITIAL
; REST LOCATIONS IN THE BIT MAP
;
; STORED IMAGE LOCATIONS:
; $9E80-$9EA8 NUB,N1,N2,N3,N4
; *********************************
;
       * = $9070
;
; NUB, N1-N4
;
CHARA  .WORD $D098,$D188,$D190
       .WORD $D198,$D1A0,0
PLACE1 .WORD $9E80,$9E88,$9E90
       .WORD $9E98,$9EA0
PLACE2 .WORD $3DD8,$3288,$3298
       .WORD $32A8,$32B8
;
       LDA $DC0E     ;TURN OFF
       AND #$FE      ; KEYSCAN
       STA $DC0E     ;  INTERRUPTS
       LDA $01       ;SWITCH IN
       AND #$FB      ; CHARA
       STA $01       ;  SET
       LDX #$00
NEXCHR LDA CHARA,X
       BEQ SCANIN
       STA $FB       ;LOBYTE
       LDA PLACE1,X  ; FROM
       STA $FD       ;  TO (1)
       INX
       LDA CHARA,X
       STA $FC       ;HIBYTE
       LDA PLACE1,X  ; FROM
       STA $FE       ;  TO (1)
       JSR COPY
       DEX
       LDA PLACE2,X  ;LOBYTE
       STA $FD       ; TO (2)
       INX
       LDA PLACE2,X  ;HIBYTE
       STA $FE       ; TO (2)
       JSR COPY
       INX
       JMP NEXCHR    ;GET ANOTHER
COPY   LDY #$00
MOVE   LDA ($FB),Y   ;GET DATA
       EOR #$FF      ;COMPLEMENT
       STA ($FD),Y   ;STORE DATA
       INY
```

```
        CPY #$08        ;DO 8
        BNE MOVE        ;TIMES
        RTS
SCANIN  LDA $01         ;SWITCH
        ORA #$04        ; I/O
        STA $01         ;  IN
        LDA $DC0E       ;RESTART
        ORA #$01        ; KEYSCAN
        STA $DC0E       ;  INTERRUPT
        RTS             ;ALL DONE
```

```
; NAME: DRAWDN
; PURPOSE: PART OF PLOT ROUTINE
;
; $FB LOBYTE PRES POSITION
; $FC HIBYTE PRES POSITION
; $FD DRAW SHAPE
; $FE HEIGHT VALUE
; *********************************
;
        * = $9490
DRAWDN LDY #$00
       LDX #$00      ;CLEAR COUNTER
       JMP DRAW1
LOINC1 INC $FB       ;INC LOBYTE
DRAW1  CPX $FE       ;CHECK IF
       BEQ DONDRA    ; DONE
       LDA $FD       ;GET SHAPE
       STA ($FB),Y   ;DRAW IT
       INX
       LDA $FB       ;MASK MSN
       AND #$07      ; OF LOBYTE
       CMP #$07
       BNE LOINC1
HIINC1 CLC
       LDA $FB       ;LOBYTE
       ADC #$39      ;CALCULATING
       STA $FB       ;
       LDA $FC       ; NEXT ROW
       ADC #$01      ;
       STA $FC       ;  LOCATION
       JMP DRAW1
DONDRA RTS
```

```
; NAME: CLRDN
; PURPOSE: PART OF PLOT ROUTINE
;
; $FB LOBYTE PRES POSITION
; $FC HIBYTE PRES POSITION
; $FD DRAW SHAPE
; $FE HEIGHT VALUE
; *********************************
;
        * = $9430
CLRDN  LDY #$00
       LDX #$00       ;CLEAR COUNTER
MASK   LDA $FB        ;MASK MSN
       AND #$07       ; OF
       CMP #$07       ;  LOBYTE
       BEQ HIINC
LOINC  INC $FB
CLEAR  LDA #$00       ;CLEAR
       STA ($FB),Y    ; BIT MAP
       INX            ;INCR COUNTER
       CPX $FE        ;DONE?
       BNE MASK
       RTS            ;ALL DONE
HIINC  CLC
       LDA $FB        ;CALCULATING
       ADC #$39       ;
       STA $FB        ; NEXT
       LDA $FC        ;
       ADC #$01       ;  ROW
       STA $FC        ;
       JMP CLEAR      ;   LOCATON
```

```
; NAME: DRAWUP
; PURPOSE: PART OF PLOT ROUTINE
;
; $FB LOBYTE PRES POSITION
; $FC HIBYTE PRES POSITION
; $FD DRAW SHAPE
; $FE HEIGHT VALUE
; **********************************
;
        * = $9400
DRAWUP LDY #$00
       LDX #$00       ;CLEAR COUNTER
       JMP DRAW
LODEC  DEC $FB        ;DEC LOBYTE
DRAW   CPX $FE        ;CHECK IF
       BEQ FINDRA     ; DONE
       LDA $FD        ;GET SHAPE
       STA ($FB),Y    ;DRAW IT
       INX
       LDA $FB        ;MASK MSN
       AND #$07       ; OF LOBYTE
       BNE LODEC
HIDEC  SEC            ;CLEAR BORROW
       LDA $FB        ;LOBYTE
       SBC #$39       ;CALCULATING
       STA $FB        ;
       LDA $FC        ; NEXT ROW
       SBC #$01       ;
       STA $FC        ;  LOCATION
       JMP DRAW
FINDRA RTS
```

```
; NAME: CLRUP
; PURPOSE: PART OF PLOT ROUTINE
;
; $FB LOBYTE PRES POSITION
; $FC HIBYTE PRES POSITION
; $FD DRAW SHAPE
; $FE HEIGHT VALUE
; ********************************
;
         * = $9460
CLRUP  LDY #$00
       LDX #$00       ;CLEAR COUNTER
MASK1  LDA $FB        ;MASK MSN
       AND #$07       ; OF
       BEQ HIDEC1     ;  LOBYTE
LODEC1 DEC $FB
CLEAR1 LDA #$00       ;CLEAR
       STA ($FB),Y    ; BIT MAP
       INX            ;INCR COUNTER
       CPX $FE        ;DONE?
       BNE MASK1
       RTS            ;ALL DONE
HIDEC1 SEC
       LDA $FB        ;CALCULATING
       SBC #$39       ;
       STA $FB        ; NEXT
       LDA $FC        ;
       SBC #$01       ;  ROW
       STA $FC        ;
       JMP CLEAR1     ;   LOCATON
```

```
; NAME: PRESPPO
; PURPOSE: TO INITIALIZE THE
; PRESENT POSITION POINTERS FOR
; EACH OF THE DISPLAYS
; TO CLEAR THE HEIGHT VALUES & SET
; THE NOTCH & NUB LOCATION POINTERS
; $9F00-9F3F ENERGY DISPLAY
; $9F40-9F7F BAND GAIN I
; $9F80-9FBF BAND GAIN II
; $9FC0-9FFF INIT EQU
; $9E00-9E3F BAND GAIN HEIGHT
; $9E40-9E5F INIT EQU HEIGHT
;***********************************
;
       * = $9300
ROW    .WORD $2647,$2F07
       .WORD $2B47,$3A47,0
PLACE  .WORD $9FC0,$9F40
       .WORD $9F80,$9F00
;
       CLD
       LDX #$00
       LDY #$00
AGAIN  LDA ROW,X     ;GET LOBYTE
       BEQ FINPRE    ;CHECK FOR ZERO
       STA $FB       ;STORE ZERO PAGE
       INX
       LDA ROW,X     ;GET HIBYTE
       STA $FC
       INX
       LDA PLACE,Y
       STA $FE
       INY
       LDA PLACE,Y
       STA $FF
       INY
       TYA           ;STORE Y REG
       PHA           ;ON STACK
       CLC
       LDY #$00
STORE  LDA $FB       ;LOBYTE
       STA ($FE),Y
       INY
       LDA $FC
       STA ($FE),Y
       INY
       CPY #$40      ;DO 32 TIMES
       BEQ NEXT      ;   (64)
       LDA $FB
       ADC #$08      ;CALCULATE
       STA $FB       ;
       LDA $FC       ; NEXT
       ADC #$00      ;
       STA $FC       ;  COLUMN
```

```
        JMP STORE
NEXT    PLA         ;GET Y REG
        TAY         ; FROM STACK
        JMP AGAIN
;
; CLEAR HEIGHT VALUES
;
FINPRE LDY #$00
        STY $FB
        LDA #$9E
        STA $FC
        LDA #$00
CLEAR   STA ($FB),Y
        INY
        CPY #$60
        BNE CLEAR
;
; SET NOTCH $ NUB LOCATIONS
; TO REST      $FF
;
        LDA #$B0
        STA $FB
        LDY #$00
        LDA #$FF
SETNN   STA ($FB),Y
        INY
        CPY #$05
        BNE SETNN
        RTS            ;ALL DONE
```

```
;NAME: SETCOLOR2
;PURPOSE: SET SELECTIVE PARTS
;         OF THE SCREEN
;ROWS 6,15,24 (BLUE)
;**********************************
;
        * = $9040
        LDY #$00
        LDA #$06     ;BLUE/BLACK
ROW6    STA $04F0,Y ;ROW 6
        INY
        CPY #$28    ;40 COLS
        BNE ROW6
;
;DOING THE SAME FOR ROW 15
;
        LDY #$00
ROW15   STA $0658,Y
        INY
        CPY #$28
        BNE ROW15
;
;DOING THE SAME FOR ROW 24
;
        LDY #$00
ROW24   STA $07C0,Y
        INY
        CPY #$28
        BNE ROW24
        RTS          ;ALL DONE
```

```
; NAME: SETMAP2
; PURPOSE: TO PUT IN THE WORDING
;
;**********************************
;
        * = $9200
        ; INIT
LETTER  .BYTE $48,$70,$48,$A0,0
        ;EQ
        .BYTE $28,$88,0
        ;SETTINGS
        .BYTE $98,$28,$A0,$A0,$48
        .BYTE $70,$38,$98,0
        ;BAND
        .BYTE $10,$08,$70,$20,0
        ;GAINS
        .BYTE $38,$08,$48,$70,$98,0
        ;NOTCHES
        .BYTE $70,$78,$A0,$18,$40
        .BYTE $28,$98,0
        ;ENERGY
        .BYTE $28,$70,$28,$90,$38
        .BYTE $C8,0
        ;SPECTRUM
        .BYTE $98,$80,$28,$18,$A0
        .BYTE $90,$A8,$68,0
        ;NUB
        .BYTE $70,$A8,$10,0
;
        ;ROWS 1,2,3
PLACE   .WORD $2250,$2398,$24C0
        ;ROWS 9,10,13
        .WORD $2C50,$2D90,$3148
        ;ROWS 18,19,22
        .WORD $3788,$38C0,$3C90,0
;
        LDA $DC0E      ;TURN OFF
        AND #$FE       ; KEYSCAN
        STA $DC0E      ;  INTERRUPTS
        LDA $01        ;SWITCH IN
        AND #$FB       ; CHARACTER
        STA $01        ;  SET
        LDA #$D0       ;ADDR HIBYTE
        STA $FC        ; FOR CHARA
        LDX #$00
        LDY #$FF
        CLD
NEXWOR  LDA PLACE,X
        BEQ KEYSCN     ;DONE IF EQ
        STA $FE        ;LOBYTE
        INX
        LDA PLACE,X
        STA $FF        ;HIBYTE
        INX
```

```
NEWCHR INY
       LDA LETTER,Y ;GET CHARA
       BEQ NEXWOR   ;DO NEXT WORD
       STA $FB
       TYA
       PHA
       LDY #$00
GETBYT LDA ($FB),Y  ;GET FROM ROM
       STA ($FE),Y  ;PUT IN BITMAP
       INY
       CPY #$08     ;COPY 8 BYTES
       BNE GETBYT
       PLA
       TAY
       CLC
       LDA $FE      ;CALCULATE
       ADC #$08     ; NEXT
       STA $FE      ;  BIT
       LDA $FF      ;   MAP
       ADC #$00     ;    LOCATION
       STA $FF
       JMP NEWCHR   ;COPY ANOTHER
KEYSCN LDA $01      ;SWITCH
       ORA #$04     ; I/O
       STA $01      ;  IN
       LDA $DC0E    ;RESTART
       ORA #$01     ; KEYSCAN
       STA $DC0E    ;  INTERRUPT
       RTS          ;ALL DONE
```

```
; NAME: SETMAP1
; PURPOSE: TO SET THE RULER LIKE
;          LINES AT ROWS 13 & 22
;
;**********************************
;
          *=$9100
        LDA #$40        ;ROW13 LOBYTE
        STA $FE
        LDA #$30        ;ROW13 HIBYTE
        STA $FF
        LDA #$3F        ;LOBYTE FINISH
        STA $FB
        LDA #$31        ;HIBYTE FINISH
        STA $FC
        JSR RULE
        LDA #$80        ;DOING
        STA $FE
        LDA #$3B        ; AS
        STA $FF
        LDA #$7F        ;  ABOVE
        STA $FB
        LDA #$3C        ;   FOR
        STA $FC
        JSR RULE        ;    ROW 22
;
; CORRECTION OF COL 0 ROWS 13,22
;
        LDA #$42        ;ROW13 COL0
        STA $FE         ; HIBYTE
        LDA #$30        ;  LOBYTE
        STA $FF
        JSR FIXL
        LDA #$82        ;SAME
        STA $FE         ; FOR
        LDA #$3B        ;  ROW 22
        STA $FF
        JSR FIXL
;
; CORRECTION OF COL31 ROWS 13,22
;
        LDA #$3A        ;ROW13 COL31
        STA $FE         ; HIBYTE
        LDA #$31        ;  LOBYTE
        STA $FF
        JSR FIXR
        LDA #$7A        ;SAME
        STA $FE         ; FOR
        LDA #$3C        ;  ROW 22
        STA $FF
        JSR FIXR
        RTS             ;ALL DONE
;
;**********************************
```

```
;
; FIX LEFT SUBROUTINE
;
;*********************************
;
FIXL   LDY #$00
       LDA #$01
CLEARL STA ($FE),Y
       INY
       CPY #$03      ;DO 3 PIXELS
       BNE CLEARL
       LDA #$00
CLEARA STA ($FE),Y
       INY
       CPY #$05      ;DO 2 MORE
       BNE CLEARA
       RTS
;
;*********************************
;
; FIX RIGHT SUBROUTINE
;
;*********************************
;
FIXR   LDY #$00
       LDA #$80
CLEARR STA ($FE),Y
       INY
       CPY #$03      ;DO 3 PIXELS
       BNE CLEARR
       RTS
;
;*********************************
;
; CALCULATE NEXT COLUMN LOCATION
; SUBROUTINE
;
;*********************************
;
NEXCOL CLD
       CLC
       LDA $FE       ;PRES POS
       ADC #$01      ;INCR LOBYTE
       STA $FE
       LDA $FF       ;HIBYTE
       ADC #$00      ;INCR ?
       STA $FF
       RTS
;
;*********************************
;
; FILL IN THE TOP OF EACH ROW
; SUBROUTINE
;
```

```
;*********************************
;
TOP     LDY #$00
        LDX #$00
        LDA #$FF        ;TOP 2 PIXELS
SETTOP  STA ($FE),Y     ; IN EACH
        INC $FE         ;  COL SET
        INX
        CPX #$02
        BNE SETTOP
        LDA #$81        ;PIXELS AT
        LDY #$00        ; EDGES SET
        LDX #$00
SET2    STA ($FE),Y
        INC $FE
        INX
        CPX #$03
        BNE SET2
        RTS
;
;*********************************
;
; RULE SUBROUTINE
;
;*********************************
;
RULE    JSR TOP
        LDA #$80        ;PIXEL AT
        LDY #$00        ; LEFT EDGE
        LDX #$00
SETL    STA ($FE),Y
        INC $FE
        INX
        CPX #$02
        BNE SETL
        JSR NEXCOL
        JSR TOP
        INC $FE
        INC $FE         ;AT BOTTOM
        LDA $FF         ;PRES HIBYTE
        CMP $FC         ;END HIBYTE
        BNE NEXT
        LDA $FE         ;PRES LOBYTE
        CMP $FB         ;END LOBYTE
        BEQ DONE        ;FINISHED
NEXT    JSR NEXCOL
        JSR TOP
        LDA #$01        ;PIXEL AT
        LDY #$00        ; RIGHT EDGE
        LDX #$00
SETR    STA ($FE),Y
        INC $FE
        INX
        CPX #$02
```

```
        BNE SETR
        JSR NEXCOL
        JMP RULE
DONE    RTS
```

```
;NAME: SETCOLOR1
;PURPOSE: SET ENTIRE SCREEN MEM SO
;         THAT BACKGROUND IS CYAN
;         AND CHARA COLOR IS BLACK
;SCREEN MEMORY $400-$7E7
;*********************************
;
       * = $9020
       LDA #$00     ;SETTING UP
       STA $FB      ; POINTERS TO
       LDA #$04     ;  SCREEN
       STA $FC      ;   MEMORY
       LDX #$00
       LDY #$00
COLOR  LDA #$03     ;CYAN/BLACK
       STA ($FB),Y  ;FILL
       INY          ; SCREEN
       BNE COLOR    ;  MEMORY
       INC $FC
       LDA $FC
       CMP #$08     ;FINISH AT $800
       BNE COLOR
       RTS
```

```
;
; NAME: MAPCLEAR
; PURPOSE: CLEAR CHARA. MEMORY
; CLEARS FROM $2000 TO $4000
; CHARA MEM ($2000-$3F3F)
;**********************************
;
     LOBYTE = $FB    ;LOBYTE OF ADDR
     HIBYTE = $FC    ;HIBYTE OF ADDR
;
       *=$9000
       LDA  #$00
       STA  LOBYTE ;SET ADDR OF
       LDA  #$20    ; CHARA MEM FOR
       STA  HIBYTE  ;  INDIR ADDR
       LDY  #$00    ;CLEARING
       LDX  #$00    ; REGESTERS
CLEAR  LDA  #$00    ;  USED
       STA  ($FB),Y ;CLEARING
       INY          ; BIT
       BNE  CLEAR   ;  MAP
       INC  HIBYTE
       LDA  HIBYTE
       CMP  #$40
       BNE  CLEAR   ;DONE AT $4000
       RTS
```

VITA

James Lee Stortz is the son of Patricia Ann Stortz and Philip John Stortz. He was born on May 12, 1961 in Covington, Kentucky. He obtained his secondary education from Waggener School in Saint Matthews, Kentucky.

Mr. Stortz entered the Speed Scientific School of the University of Louisville in September 1979, majoring in electrical engineering. He received the Bachelor of Science Degree in May of 1984 and the Master of Engineering Degree in December 1985. He is a member of Triangle Engineering Fraternity. He has currently attained a full-time position with Texas Instruments located in McKinney, Texas.